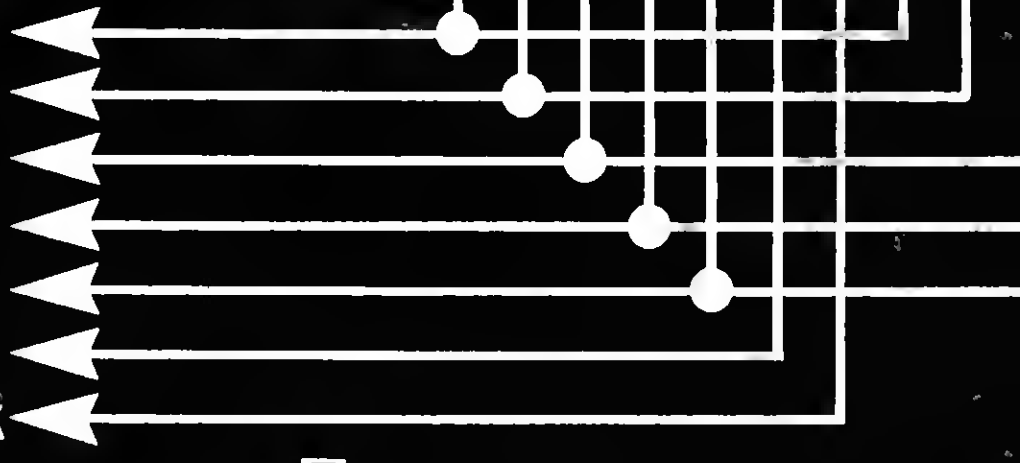
Catalog No. 26-2103

TRS-80 micro computer technical reference handbook



Radio Shack

CUSTOM MANUFACTURED IN U.S.A. BY RADIO SHACK A DIVISION OF TANDY CORPORATION

TRS-80 micro computer technical reference handbook

Radio Shack®
A DIVISION OF TANDY CORPORATION
One Tandy Center
Fort Worth, Texas 76102

FIRST EDITION
FIRST PRINTING – 1978

Printed in the United States of America

Preface

We'd like to share a couple of important points before you get into this book.

1. This Book was written for the technical person, by a technical person. It was not written to educate the average owner of a TRS-80 Micro-computer. If you do not know what Hex means . . . or how a NOR gate differs from a NAND gate . . . you are not prepared to repair your Computer. (You need some solid digital logic training; and this Book won't give it to you.)

   If you are a technical person, but have never played around with microprocessors, dive right in! A computer is nothing more than a fast idiot. Don't let the "idiot" intimidate you. All of the familiar logic is there; there just may be more of it than you're used to seeing all in one place.

   If you are a hobbyist who can convert Hex to Decimal in the blink of an eye and you've entered 16K BASIC languages using front panel switches, then this book will probably appear longer than it needs to be. All the information you seek is here; you will just have to turn more pages to find it.

2. We've incorporated a lot of technical information that can lead you right into the heart of the TRS-80 hardware. But we don't want you to infer from this that we can make available engineering design services for your hardware/circuit ideas. We'd like you to remember that any work you do on your TRS-80 "voids the warranty". And, we will not obligate ourselves to repair or correct owner-modifications.

   So, once you open up the cabinet, you're on your own.

This book has been fun to write and we hope you have fun reading and using it.

May all your logic and software be glitch-free!

Table of Contents

Introduction

Armed with only a Schematic, attacking a TRS-80 Computer may at first seem to be the ultimate exercise in futility. But that's where we come in. This book has been written with those kinds of feelings in mind (matter of fact, the writer went through many of the feelings of trepidation as he was first asked to trouble-shoot and repair an early TRS-80 . . .). You may know what "CPU" stands for. You may even have some knowledge of how a microprocessor system works.

After a while, you find you have problems. You know a 4K RAM needs 12 address lines. But you only find 7. You know a computer keyboard gives you ASCII. Yet, you find the Keyboard is shorting out an address line to a data line. You know how a TV typewriter scrolls characters on the screen, and you do find the video memory wired to do the job. But where is all the hardware to make the display scroll?

You know what a NAND gate symbol looks like and you also know what an OR gate symbol looks like. But some sadist has gotten all his symbols backwards. A NAND gate is shown like an OR gate, and the OR gate looks like a NAND gate. To top it all off, the power supply consists of two large rectangles with transistors and resistors sticking out of them; and the only voltages shown are the resulting outputs.

Welcome to the wonderful world of computer electronics!

Now admit it . . . you don't *really* know your ASCII from a scroll when it comes to the TRS-80 Computer . . . and so once again we get back to this book. Join the crowd; line up and let's all learn together. Grab this book, the Schematics and together we'll attack that TRS-80 with soldering iron smoking.

The purpose of this book is to give you a practical knowledge of system operation a'la TRS-80. This book will show you why there are only seven address inputs to a 4K RAM. We'll show you when the Microprocessor inputs data from the keyboard, the CPU thinks the Keyboard is a memory, of all things! You'll learn how the CRT screen is scrolled. (And you might even learn to appreciate the backward symbolization!) As far as the Power Supply is concerned, you might find that it's not nearly as complex as you thought. So, grab your Schematics and let's take a tour of the TRS-80 Computer.

Let's start off easy . . . with a . . .

System Block Diagram

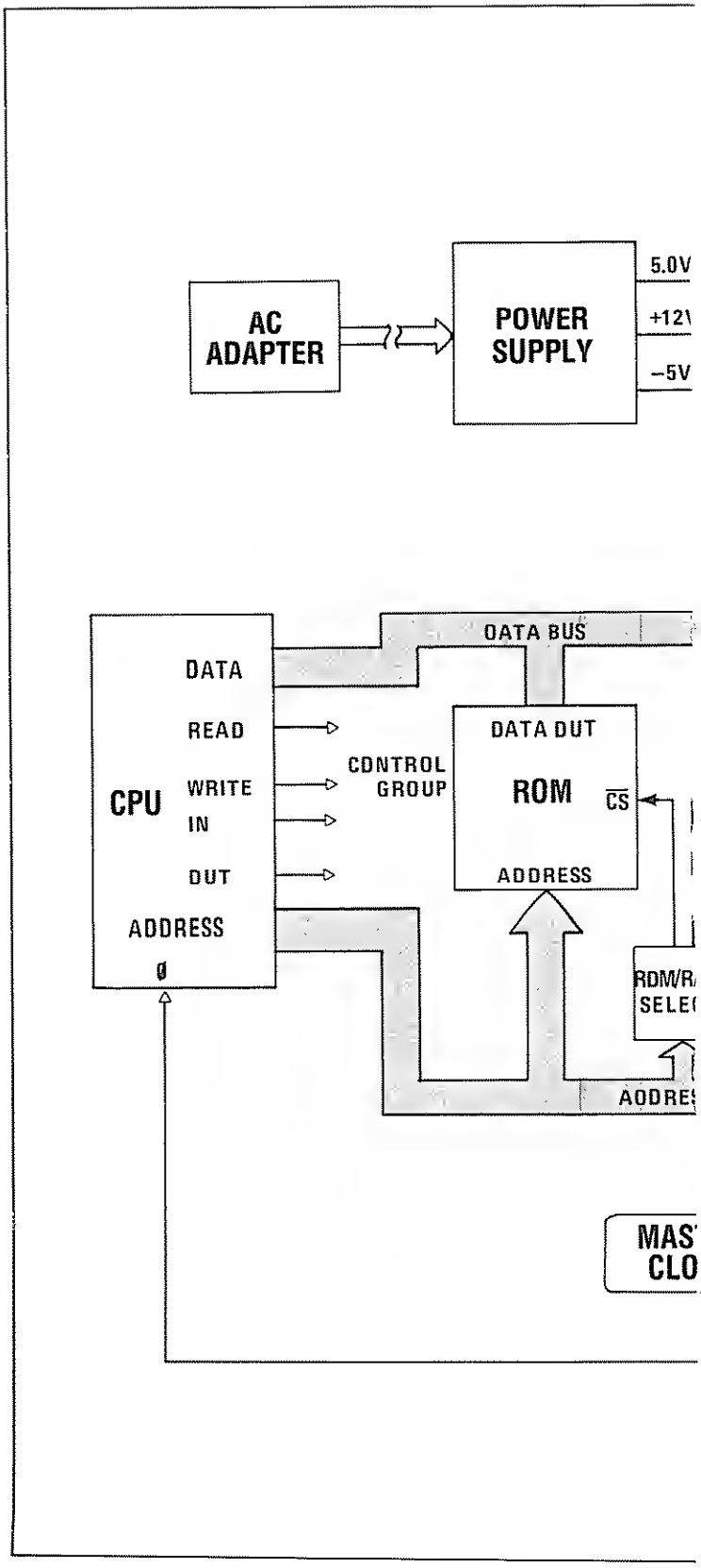
The 8Ø integrated circuits contained in the TRS-80 can be broken down into about 1Ø major sections. Figure 1 shows these sections as they relate to other sections. The heart of the system is definitely the CPU (Central Processing Unit). You might consider the CPU as being a very dumb calculator circuit. It may be dumb, but it's a fast dummy. Most of the leads on the CPU are data lines and address lines. The CPU tells the address bus where the data it wants is located. and the data bus is a good place for the information to come back to the CPU. The address lines are outputs from the CPU. They never receive data or addresses from other sections. The data lines on the other hand can give or receive data.

ROM

If the CPU has to be the heart of the system, the ROM (Read Only Memory) could be considered the brains. The ROM tells the CPU what to do, how to do it and where to put it after it's done. Without the ROM, the CPU would just sit there and oscillate. When power is first applied to the system, the CPU has just enough smarts to output an address to the ROM that locates the CPU's first instruction. The ROM shoots back the first instruction and then the two really start communicating. In less than a second, the CPU, under ROM supervision, performs all the housekeeping necessary to get the system alive and a "READY" flashes on the screen.

If the CPU misses that first piece of ROM data, then it may go bananas. It may tell the ROM that it is ready to load a tape so the ROM tells it how to do that. The Tape Recorder turns on. But since the CPU is now playing games in the video memory, who cares about the tape? The CPU operates at about 2 MHz; therefore, digital screw-ups seem instantaneous.

Remember that the CPU is the work horse and the ROM is the boss. The ROM tells the CPU how to do it, when to do it, and where it put it.



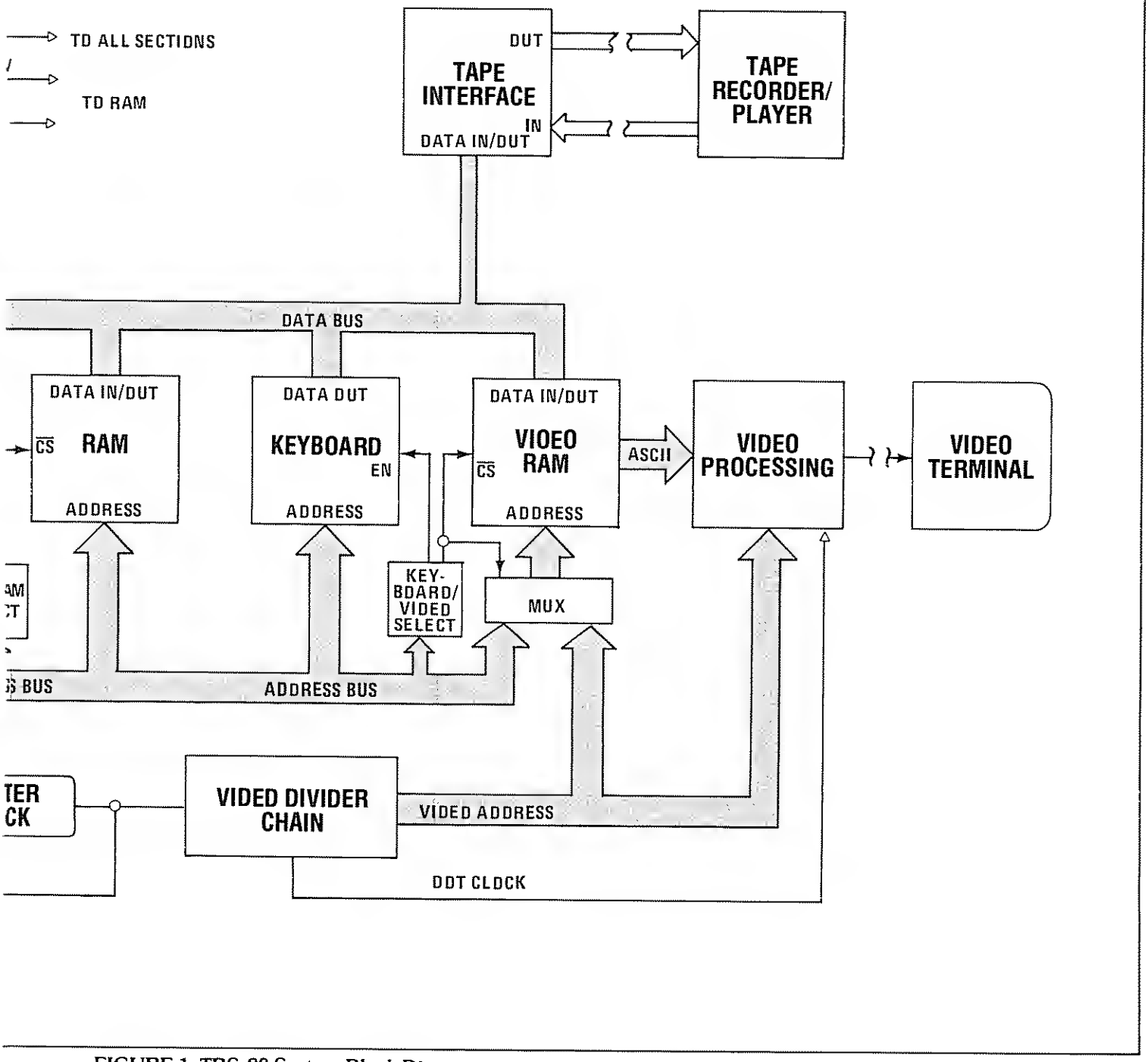


FIGURE 1. TRS-80 System Block Diagram

RAM

The next major section in Figure 1 is the RAM (Random Access Memory). This memory is where the CPU may place data it doesn't need until later. The RAM is also the place where the programs are kept. If you tell the computer to count to 10,000, then the CPU stores your instructions in the RAM. If you tell the Computer to do it *now*, here is what happens:

The CPU tells the ROM someone wants in. The ROM tells the CPU to go to the Keyboard and find out who. The CPU finds out, tells the ROM that it's **The** boss. The ROM tells the CPU to find out what he wants. The CPU tells the ROM that the boss wants *us* to RUN. The ROM tells the CPU to go to RAM and find out what the boss wants done. The CPU says the boss wants to count to 10,000. The ROM tells the CPU how to do it. After it's done, the ROM tells the CPU to find out what to do with it. The CPU informs the ROM that the 10,000 has got to go on the Display and must be saved. The ROM tells the CPU how to put it on the Display and then tells it to store the 10,000 somewhere in RAM; but it had better remember where it is. The CPU tells the ROM that the job is done. The ROM tells the CPU to monitor the Keyboard in case the boss wants something else.

The CPU looks to the ROM for instructions. The CPU then follows the ROM's instructions and looks to the Keyboard, then the RAM. In all cases, the CPU applies address locations to the ROM, RAM and Keyboard. The data lines are then checked for input data that corresponds to these address locations. In case of an output from CPU to RAM, the CPU selects the address, puts data on the data lines and then instructs the RAM to store the data that is on the data lines.

Notice that **only** the CPU communicates with all other sections. If the CPU is told by ROM to store something from ROM into RAM, the CPU can't make the RAM receive ROM data directly. Instead, the CPU takes the data from ROM and then sends it to RAM. The CPU must act as intermediary between the two. The reason for this is that the CPU is the only section that can address locations and pass data to all other sections.

Keyboard, Video RAM And Video Processing

The Keyboard section is not necessary as far as the CPU is concerned, but it is very necessary for the operator (that's you!). The Keyboard is our method of making known our instructions to the CPU. The opposite is true for the Video RAM. In this case, the CPU wants to tell us it needs data or it may want to show us the result of a complex calculation. So, the request for more information or the result is stuffed into the Video RAM. Anything in Video RAM is automatically displayed on the Terminal. The Video Processing section handles this. Data in the Video RAM is in ASCII. Converting ASCII into the alphanumeric symbols we recognize is the job of the Video Processor. A ROM contains all of the dot patterns. The ASCII locates the character pattern, and the Video Processor sends it out to the Terminal.

Video Divider Chain

Composite video going to a video terminal is extremely complex. Aside from the video signal, there is the horizontal and vertical sync. These signals must be very stable and be outputted in the correct sequence. The CPU is busy enough as it is, so the Video Divider Chain handles the TV work. It generates the sync signals and addresses the Video RAM in a logical order so that the Video Processor can handle video data efficiently. Notice the block under the Video RAM labeled MUX. This is short for Multiplexer.

It acts somewhat like a multipole, multiposition switch. When the Video Divider Chain is in control, the MUX is switched so that only addresses from the divider chain are directed to the Video RAM. The CPU may need to read or write data into the Video RAM. If so, the MUX is switched so that the CPU has control over Video RAM's address. After the CPU is finished, the addressing task is reassigned to the Divider Chain.

The Memory Map

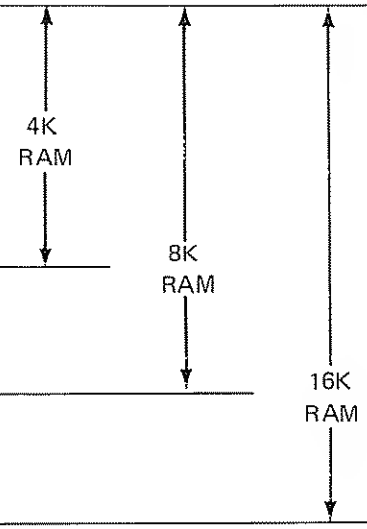
Maybe you've asked the question "Which output port is the display?" The answer is that the TRS-80 does not use an output port for the display. The TRS-80 is memory-mapped. In a memory-mapped system an address will define and select all other subsections.

Figure 2 shows the Memory Map for a Level I TRS-80. From memory locations ØØØØ to ØFFF, the Level I ROMs are present. The Keyboard is located from address 38ØØ to 38FF. The Video Display is located from 3CØØ to 3FFF. The RAMs start at 4ØØØ and, depending on how much RAM is in the system, can extend down to address 7FFF.

As we told you before, upon power-on, an address location is outputted by the CPU requesting information from the ROMs. Since the ROMs are located at addresses ØØØØ to ØFFF, the CPU will be outputting addresses in this area. If the CPU needs some kind of keyboard data, it will output addresses 38ØØ through 38FF and see if anything is in this "memory" location. If the CPU wants to show the programmer something on the display, all it has to do is address the Video Display section of the map and store data in these locations. Something to remember: the Video Display shows what is in memory locations 3CØØ through 3FFF.

Notice memory locations 4ØØØ to 41FF. At address 4ØØØ RAM starts. But, part of the RAMs are used by BASIC as general housekeeping memory locations. Hence, the user accessible RAM actually starts at address 42ØØ.

HEX ADDRESS	DESCRIPTION OF CONTENTS/USAGE
0000 To 0FFF	Level I ROMS
1000 To 37FF	Not used
3800 To 38FF	Keyboard
3900 To 3BFF	Not used
3C00 To 3FFF	Video Display
4000 To 41FF	RAM Used by BASIC LEVEL 1
4200 To 4FFF	Useable RAM starts here
5000 To 5FFF	RAM
6000 To 7FFF	RAM
8000 To FFFF	Not used



NOTE: Map not drawn to scale.

FIGURE 2. Level I Memory Map

Theory Of Operation

System Clock

The System Clock is shown on Sheet 2 of the fold-out Schematics at the back of this book. Y1 is a 10.6445 MHz, fundamental-cut crystal. It is in a series resonant circuit consisting of two inverters. Z42, pins 1 and 2, and 3 and 4, form two inverting amplifiers. Feedback between the inverters is supplied by C43, a 47 pF capacitor. R46 and R52 force the inverters used in the oscillator to operate in their linear region.

The waveform at pin 5 of Z42 will resemble a sine wave at 10.6445 MHz. The oscillator should not be measured at this point, however, due to the loading effects test equipment would have at this node. Z42, pin 6, is the output of the oscillator buffer. Clock measurements may be made at this point. The output of the buffer is applied to three main sections: the CPU timing circuit, the video divider chain, and the video processing circuit.

CPU Timing

The Z80 microprocessor needs a single phase clock source for operation. The 10 MHz signal from System Clock is applied to Z56, a standard ripple counter, which is used as a divide-by-6 counter. The resulting signal at Z56, pin 8, is a little over 1.774 MHz. The signal is applied to the input of buffer Z72, pin 12. Pin 11 of Z72 is attached to pin 6 of the Z80 microprocessor. R64 pulls up pin 11 of Z72, and insures a rapidly increasing rise time for the clock signal.

Notice that pin 15 of Z72 is tied to ground. Since pin 15 is the enable input to this part of Z72, pins 12 and 11 will always be active.

Notice also pins 7 and 6 of Z56. These two pins enable the clear function for the counter. When one or both of these pins is low, the counter operates normally. When high, the input forces the counter into its clear or reset state.

Z42, pins 9 and 8, are used to disable counter Z56 during automatic testing at the factory. R67 pulls Z42's input to V_{CC}, which causes pin 8 to stay at a logical low. During testing, pin 9 of Z42 may be pulled low, making pin 8 high, which disables and clears Z56.

NOTE: You might also find early Board versions ("A" Boards for example) where pins 6 and 7 of Z56 are tied directly to ground.

Power-Up-Clear and System Reset

As mentioned in the Block Diagram discussion, upon power-on the CPU accesses a known address in the ROM for instructions. The circuitry which causes the starting address output is shown just below the microprocessor clock divider.

Z53 is a 2-input, quad NAND gate. (Note that Z53 is drawn as an inverted input OR gate.) When power is first applied to the system, C42 is at 0 volts. R47 is tied to V_{CC} and starts charging C4 at a known rate. While C4 is charging, and before the voltage exceeds the logical 1 level for Z53, pin 11 outputs a high. This high is inverted by Z52, pins 11 and 10, and a low is applied to pin 26 of Z40.

A low at this input forces the microprocessor to output the starting address 0000 on its 16 address lines. When C42 charges up past about 1.4 volts, Z53, pin 11, goes low, which causes Z52, pin 10, to go high. The CPU is now out of

its reset state, and will start executing instructions from the ROM, starting at address 0000. Notice that the only time pin 26 of the CPU is ever low is a few milliseconds after power is applied. Once C42 charges up past the logical "1" level, pin 26 stays high until C42 is discharged when power is removed.

Why is Z53, a NAND gate, drawn as an OR gate? Notice that pin 11 is high only when either of the inputs are low. The NOT circles at the input immediately tell you that this gate is looking for a signal that is low to cause an output that is high. Had the gate been drawn "correctly", then it would not have been so obvious that the output is active when high. This "functional" type of logical symbolization is used throughout the schematics.

Directly above the power-up circuit, there is a similar circuit. S2 is the Reset switch located on the left side of the Board. Although there is a power-on-delay type circuit at the input of this network, it is not used as such. Notice that C57 is smaller than C42. Hence, in a power-up "race", C57 would charge up faster than C42. Assume that C57 is charged. Also assume that pin 2 of Z53 is high. This means that Z53, pin 3, will be low and Z37, pin 13, will be high. With pin 17 of the CPU held high, everybody is happy. If S2 is pressed, C57 will discharge through the switch. The resulting low is applied to pin 1 of Z53 and pin 3 goes high. Z37, pin 13, is then forced low. A low at pin 17 of the CPU forces the microprocessor to restart at address 0066. When S2 is released, R65 begins to charge C57 until a logical high is applied to pin 1 of Z53. At this time, pin 17 of the CPU goes back high and the CPU starts executing instructions from address 0066 in the ROMs.

S2 is used to get the microprocessor back on the right road when it is "lost". This switch forces the CPU toward a known address to enable it to get on the right track. An example of a lost CPU would be during a bad cassette load attempt. If a cassette is loading and suddenly there is missing information on the tape (caused by dirt or age), the recorder may never stop. S2 can then be pressed, which directs the CPU out of the cassette load routine and back into its ready mode.

The output at pin 18 of Z40 is called "Halt". In Level I BASIC this output should never be low. It goes low only when a software halt instruction is encountered by Z40. In theory, this instruction is not included in the ROMs. But you might find pin 18 held low because Z40 thought it was told to halt. It could be due to some data malfunction, or the CPU is lost and is playing around with display data instead of ROM data. In a case like this, S2 is not effective in bringing the CPU home, because Z53 is latched up. About all you can do is shut the Computer down and try again.

Notice that Z53, pins 11 and 3, are also tied to Z37, pins 2 and 3. Z37, pin 1, is an output line labeled SYSRES* (System Reset Not). It is normally-high and only goes low during power up (Z53, pin 11, causes this), or when S2 is pressed (Z53 pin 3, causes this). SYSRES* is used by the expansion interface and is not used by the TRS-80 in Level I BASIC.

One last thing to mention about these two circuits: When you turn off power to the TRS-80 because of a lost CPU, wait at least 10 seconds before you reapply power. If you do not wait, C42 may not discharge completely and the CPU may not go back to address 0000 during a restart. By waiting, C42 will discharge and upon power-up, the system will start at the correct ROM location.

WAIT, INT* and TEST

These three inputs to the CPU are pulled up to V_{CC} through resistors. Since they are active low, you may not have any use for them. But you should know what they are for.

The WAIT input, pin 24 of Z40, will slow the CPU down if there are slow memories it must access. If this line goes low, the CPU will go into

a wait status until it goes back high. Once high, the CPU continues with the operation. For example: Assume you have a memory system that takes 1ØØ microseconds before addressed data can be guaranteed to be present at the output. When the memory logic sees that the CPU wants data, it will make the WAIT line low. After 1ØØ microseconds, the logic will make the WAIT pin high, and the CPU will input the data.

The INT (Interrupt Request) is at pin 16 of Z4Ø. When low, this input will force the CPU into an interrupt request section of the memory. It would then perform some instruction associated with the interrupt. An example of this use would be as follows: Assume there is a door on the back of the TRS-80 that should always be closed. There is a switch connected to the door such that when opened, the switch contacts are shorted. The switch is connected to ground and to pin 16 of Z4Ø. If the door is opened, the Computer stops what it is doing and prints on the screen "Close Door." The CPU would be interrupted, and it would henpeck you until you closed that door! As you can see, pin 16 is tied to V_{CC} through a resistor and is not used. However, it is used with the TRS-80 Expansion Interface.

The TEST input may be quite useful in your troubleshooting. Pin 25 of Z4Ø is labeled †$\overline{8USRO}$ (8us Request). When this pin is brought low, it will force the data, the address and the control lines into the disabled or floating state. Although it is not used by the TRS-80 in normal operation, it is quite useful when someone wants to "shut down the CPU". We'll talk about this input when we discuss the Control Logic Group.

†Note that we use two types of signal names. If a bar is placed over a signal name, as in $\overline{BUSRO}$, then read it as "NOT BUSRO" (which means it is an active low signal or pin). BUSRO* also reads "NOT BUSRO", and refers to an active low signal or pin. The reason the asterisks are used instead of the bar, is because computer printouts write asterisks better than bars. Throughout this book we use both methods of naming signals (we'll use the method which appears on the Schematic [which typically is the manufacturer's name - label]).

CPU Address Lines

There are system outputs of the microprocessor labeled AØ through A15 that start the address bus. Since these lines must go to ROM, RAM, the Keyboard and the Video RAM, they must be buffered for two reasons.

First, the buffers must be able to supply the address bus with proper logical levels. The microprocessor cannot supply the current necessary to drive all of the sections connected to the address bus, and buffers are needed for current gain.

Secondly, it may be necessary to switch off the address bus. For example, if an Expansion Interface is connected to the bus, it may be necessary to address RAM in the main unit for a data transfer. Therefore, there must be some method to take the CPU off the data bus. The buffers are tri-state devices. This means they will either act as buffers or as opened switches.

Z38, Z39 and part of Z22 and Z55 are the address line buffers. Notice that in Z38 and Z39 there are two sections of buffers. The first section contains four buffers and the second section contains only two buffers. Each section is controlled by a single pin. The first is controlled by pin 1 and the second by pin 15. When these control pins are at a logical low, the buffers are enabled and will operate normally. When the control pin is at a logical high, the buffers are disabled, and will show a high impedance from input to output. The signal that controls the address buffers is labeled ENA8LE* and is sourced at Z52, pin 4. Pin 3 is the input for control line inverter, Z52, and is tied to the TEST* line. Notice that R58 keeps this line pulled high. Thus, the address buffers' control line will always be at a logical low; and therefore, operating as buffers. If TEST* is shorted to ground, the address buffers will be disabled. This feature could be very useful in troubleshooting.

CPU Data Bus

The data bus is buffered like the address bus, except for one area. Notice that there are only eight data lines at the CPU, labeled DØ through D7. But there at 16 buffers. Remember that the CPU must receive data as well as send data. The address lines are strictly CPU outputs, while the data lines are inputs and outputs. Therefore, there must be two sets of buffers for the data line. One set handles CPU output data while the other set takes care of the CPU input data.

The output data buffers consist of Z75 and one section of Z76. The input buffers consist of one section of Z55 and the last section of Z76. Notice that the input and output buffers are connected "head to toe". This could cause problems if both were on at the same time! The control inputs to the output buffers are all connected together on the line labeled DBOUT*, and are in turn tied to Z53, pin 6. Likewise, the input buffers' controls are tied together on the line labeled DBIN*, and are connected to Z53, pin 8. DBOUT*, is tied to pins 9 and 1Ø of Z53, the gate which generates DBIN*.

As you can see, Z53, pin 6, is the major source of input or output data control. If pin 6 is high, DBOUT* is high and DBIN* is low. Therefore, the input buffers are enabled and the output buffers are disabled. If Z53, pin 6, is low, DBOUT* is low and DBIN* is high. In this case, the output buffers are enabled and the input buffers are off.

Pin 4 of Z53 is tied to TEST*. If TEST* is grounded, not only will we disable the address buffers, but we will also cause pin 6 of Z53 to go high. Thus, the data output buffers will be off, robbing the CPU's control over the data lines. Since DBIN* is now held low, the input data buffers would be active. But, this would not cause any problem since the address bus from the CPU has been disabled.

When TEST* is left alone, it is held high. If pin 21 of the CPU (the Memory Read output) is high, Z53, pin 6, will be low. The low causes DBOUT* to be low and DBIN* to be high. Therefore, the CPU is outputting data; and the buffers are switched accordingly. When pin 21 of Z4Ø goes low, Z53, pin 6, will be high. We now have almost the same condition as if TEST* went low. DBOUT* is high and DBIN* is low but the address buffers are still enabled. The data buffers are now ready for the CPU to accept data.

CPU Control Group

OK, we now know how the CPU accesses the address bus. We know the data bus is used to gather data into the CPU or pass data out of the CPU. What we do not know at this point is how the CPU stores data in a memory or how it tells the ROM or RAM that it is ready to receive data. The CPU control group performs this task. These signals are: RD, WR, OUT and IN.

RD (Read)

RD is Read control. This signal, when activated, will tell other sections that the CPU is ready to accept data. RD is generated at Z23, pin 6. Pin 5 is connected to pin 21 of Z4Ø, the $\overline{RD}$ (Read) output. Pin 4 of Z23 is tied to pin 19, $\overline{MREQ}$ (Memory Request), of the CPU. Therefore, when pins 19 and 21 of Z4Ø go low at the same time, an RD output is generated. Notice the backward symbol for an OR gate. It's drawn as an AND gate. When we get $\overline{MREQ}$ and $\overline{RD}$, *then and only then* will we get RD. We're looking for two lows on the input for a low output.

WR (Write)

WR is Write control. This signal, when activated, will tell other sections that the CPU is ready to write data into one of the memory locations. WR is generated at Z23, pin 11. Pin 12 of Z23 is connected to $\overline{MREQ}$. Pin 13 of Z23 is tied to $\overline{WR}$ (Memory Write), which is pin 22 of Z4Ø. When we get a low at the $\overline{MREQ}$ output and a low at the $\overline{WR}$ output, *then and only then* will we get a low at WR.

OUT (Output)

OUT is Output control. This signal, when activated, will enable circuitry to perform the cassette save functions. It also is used to control data movement from the TRS-80 to the Expansion Interface. OUT is generated at Z23, pin 3. Pin 1 of Z23 is tied to the $\overline{WR}$ output on the CPU. Pin 2 of Z23 is tied to $\overline{IORQ}$ (Input/Output Request) which is pin 2Ø of the CPU. When we get a low at $\overline{WR}$ and a low at $\overline{IORQ}$, *then and only then* will we get a low at OUT.

IN (Input)

IN is Input Control. This signal, when activated, will enable circuitry to perform the cassette load function. It also is used to control data movement from the Expansion Interface to the TRS-80. IN is generated at Z23, pin 8. Pin 1Ø of Z23 is connected to $\overline{IORQ}$. Pin 9 of Z23 is tied to $\overline{RD}$. When we get a low at $\overline{RD}$ and a low at $\overline{IORQ}$, *then and only then* will we get a low at IN.

Control Group Bus

The Control Group must be buffered for use by the different sections. Also, the bus may need to be switched off at some time. Therefore, part of Z22 is used to buffer the Control Group. Tri-state control at pin 1 is tied to the address bus control, and ENABLE* will affect the status of the address and the control group bus in the same manner.

Address Decoder

As shown in Figure 2, the TRS-80 is memory mapped. Therefore, the address Ø1AC (in HEX) is in the ROM part of the map. Address 38ØA is in the Keyboard area and 3CAA accesses the Video Display RAMs. Since the data and address buses are connected in parallel to all the sections, there must be some method to determine which section is being accessed. A decoding network monitors the higher order address bits and selects which "memory" the CPU wants to use.

The address decoder is so important to the operation of the system that it has been redrawn in Figure 3 (on the next page). Keep your Schematic handy since there are signals shown in Figure 3 that need to be sourced or traced.

The address decoder uses six bits AlØ through A15 are needed plus RD* and RAS* (ROW Address Select). A15 is the most significant bit of the address bus. Let's combine the six high order bits and add a couple more, so that we have two hex digits:

A15 A14 A13 A12 A11 A10 A9 A8

A12 through A15 form the most significant hex character. A8 through A11 form the next most significant hex character. A8 and A9 are the two bits we had to add to complete that last hex character. Now let's break down part of the memory map into hex and binary (see chart below).

Notice in the breakdown that we could use the two most significant digits of the hex code in the decoding scheme and handle the selection of all the memories. In the binary columns, you can see that instead of using two hex digits, which is eight binary lines, we can ignore two bits and use only six binary lines. A dotted line separates the two unused bits from the six that we'll use.

	A15	A14	A13	A12	A11	A10	A9	A8	
From: Hex 0000	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Level I ROMs
To: Hex ØFFF	Ø	Ø	Ø	Ø	1	1	1	1	
From: Hex 3800	Ø	Ø	1	1	1	Ø	Ø	Ø	Keyboard
To: Hex 38FF	Ø	Ø	1	1	1	Ø	Ø	Ø	
From: Hex 3C00	Ø	Ø	1	1	1	1	Ø	Ø	Display RAMS
To: Hex 3FFF	Ø	Ø	1	1	1	1	1	1	
From: Hex 4000	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	4K RAM
To: Hex 4FFF	Ø	1	Ø	Ø	1	1	1	1	
From: Hex 4000	Ø	1	Ø	Ø	Ø	Ø	Ø	Ø	16K RAM
To: Hex 7FFF	Ø	1	1	1	1	1	1	1	

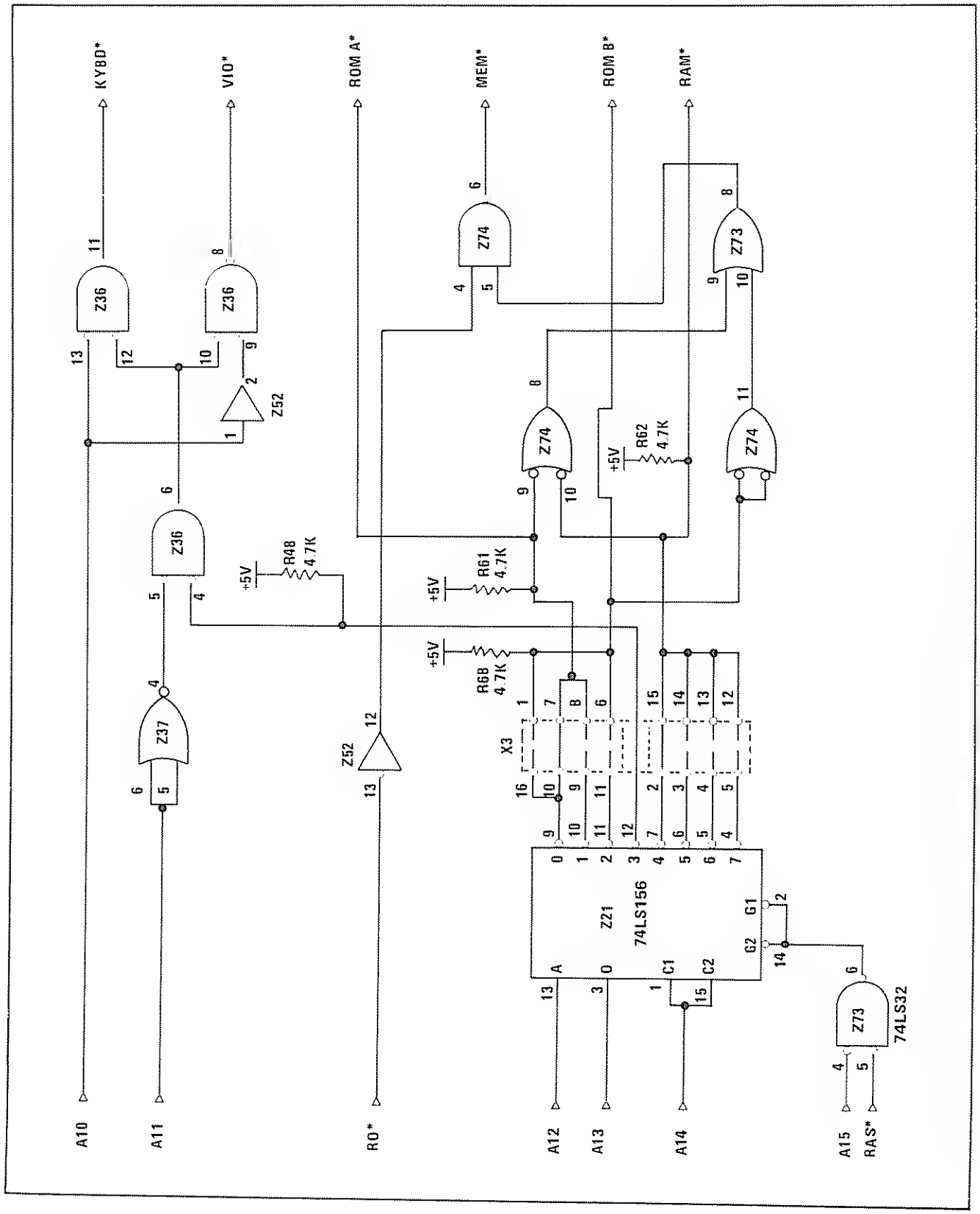


FIGURE 3. Address Decoder

Now, look at Figure 3 and you'll see that bits A12, A13 and A14 are connected to Z21, a dual, 2-input to 4-line decoder/demultiplexer. The C1 and C2 inputs are connected in such a way as to make Z21 into a 3-input to 8-line decoder. The G1 and G2 inputs to Z21 are chip enables. As shown, when these inputs are at a logical Ø, Z21 is active. When high, Z21 is disabled and none of its eight outputs are low. The G-enables are controlled by OR gate Z73, pins 4, 5 and 6. Pin 4 is tied to A15, the most significant bit of the address bus.

Notice in the memory map breakdown that A15 is always low when addressing the various memories. Z73, pin 5, is tied to RAS* (Row Address Select). Go back to the large Schematic, Sheet 1, and find $\overline{\text{MREQ}}$ at pin 19 of the CPU. As stated earlier, $\overline{\text{MREQ}}$ only goes low when the CPU needs or wants to output memory data. Follow pin 19 down to Z72, pin 4. This buffer sources RAS* and it is the same signal as $\overline{\text{MREQ}}$.

Back to Figure 3. When A15 and RAS* are low at the same time, a low will be outputted by Z73, pin 6. This low will enable Z21. When Z21 turns on, one of its outputs will go low, depending on the status of A12, A13 and A14. For example, if these three inputs are at logical zero, pin 9 will go low. If all three inputs are high, pin 4 will go low. You might consider A12 through A14 as supplying an octal address to Z21. Since there are eight states in an octal code, then there could be one of eight lines selected (Output Ø through Output 7).

We can sum up Z21's function quite simply: It decodes the most significant digit of the hex address. Using Z21 and the last two bits, A11 and A1Ø, we can define any one of the four "memories" available to the CPU in Level I.

Address Decoder Programming

Attached to the outputs of Z21 is X3. X3 is called a "DIP shunt" and it is installed in the PCB position Z3. A DIP shunt is like a shorting bar array, except the bars may be broken. By breaking some bars and leaving others intact, the address decoder is programmed to reflect the amount of RAM or ROM the CPU has available for use. In Figure 3, X3 is shown with six broken shorting bars. We will use this configuration in our discussions.

ROM Decoding

When the CPU needs instructions on how to perform a certain task, it must access ROM. ROM Decoding is performed as follows: The CPU needs a memory, so RAS* will go low. The address for ROM starts with hex Ø, so A15, A14, A13, and A12 go low. Z21 becomes active due to the low at A15 and RAS*. Pin 9 of Z21 goes low. Follow pin 9 through the shorted bar at X3 pins 1Ø and 7, past the pull-up resistor R61 and out to ROMA*. If you find ROMA* on the large Schematic, you'll see it goes to ROM A, (Z33, pin 2Ø.) This pin is the $\overline{\text{CS}}$ (Chip Select) and it is active low (as the inverting circle on pin 2Ø shows). Z33 turns on, which means that its output becomes active (Note: The ROM's outputs are tri-stateable like the buffers. When $\overline{\text{CS}}$ goes low, the ROM outputs will switch from a high impedance or off-state to an on-state. When on, the outputs will go low or high depending on the data in the ROM at the address specified.)

We got the address applied to ROM A and we got ROMA* to go low, so ROM A is turned on. But now we need to insure a data path is opened so that we can pass data from ROM to CPU. Notice in Figure 3, ROMA* is also attached to pin 9 of NAND gate Z74. A low on pin 9 will cause a resulting high at pin 8. Z74, pin 8, is tied to Z73, pin 9. Z73, pin 8 passes a high to Z74, pin 5. Z74, pin 4, is tied to RD*, part of the CPU control group. Since the CPU is trying to read data from ROMs, RD* will be low. Pin 4 of Z74 will then be high, because RD* is inverted by Z52, pins 13 and 12.

OK, we know pins 4 and 5 of Z74 are high, so that makes pin 6 low. This low is MEM*. If you

find MEM* on the big Schematic, you will notice it controls the ROM/RAM buffers. The outputs of the buffers are tied to the data bus. We now get ROM data onto the data bus. Has it got a way to get to the CPU? Yes, it does. Remember that $\overline{RD}$ is low because the CPU is in a Memory Read cycle. Since this is so, DBIN* is low and DBOUT* is high. The low at DBIN* enables the CPU's input data buffers and ROM data is available for the CPU.

Keyboard Decoding

The Keyboard is located from address 38ØØ to 38FF. The Keyboard is memory, so RAS* will be low. A15 is low because we are generating address codes under 8ØØØ. Looking at our binary location for the Keyboard, we find A14 low, A13 and A12 high. With this input combination, Z21 will be active and pin 12 will be low (Output 3). Pin 12 is tied to Z36, pin 4. According to the breakdown, A11 is high. Z37, pin 4, outputs a low to Z36, pin 5. The "incorrectly drawn" OR gate tells us we need both inputs low for a low output. We've got it, so pin 6 of Z36 is also low. Pin 6 of Z36 is tied to pins 12 and 1Ø of Z36. Checking on the status of A1Ø, we find it listed as being low during a Keyboard address output. Since Z36, pins 12 and 13 are low, we'll get a low at pin 11. KYBD* is generated at this pin.

Finding KYBD* on the big Schematic, you'll see it goes to the enable inputs of the data buffers for the Keyboard. The lower order address lines are tied to one end of the keyboard matrix, while the other end of the matrix is tied to the data bus, through the buffers. If a key is pressed, an address line will be "shorted" to a data line. Assume for now that this scheme works. We'll analyze the Keyboard later. The DBOUT* and DBIN* signals are switched the same way as if we had a ROM select. Therefore, Keyboard data will get to the CPU's data bus for processing.

Video Display RAM Select

In the binary breakdown for the Memory Map, you will notice that the binary out for the Video RAM address is almost the same as the Keyboard except for bit A1Ø. Z21, pin 12, will output a low to Z36, pin 4. Since A11 is still high, Z37, pin 4 will supply a low to pin 5 of Z36. Therefore, pin 6 of Z36 is low, just as if a keyboard was selected. Since A1Ø is now high, Z36, pin 11 is high and KYBD* is not active. But Z36, pin 1Ø is low and so is pin 9 (due to the effects of inverter Z52, pins 1 and 2). Thus, Z36, pin 8 goes low and we have caused VID*, the Video RAM select, to become active. Assume for now that VID* does select the Video RAMS. We'll discuss what it does and how it does it later.

4K RAM Decoder

As shown on the Memory Map, the addresses which select RAM extend from hex 4ØØØ to 4FFF for 4K. The binary breakdown lists the state of A15 as a Ø, of course. A14 is high and A13 and A12 are low. We are still accessing memory, so RAS* is low. Hence, Z21 will be active and output 4 will be low (pin 7). DIP shunt X3 passes this low through pins 2 and 15, and it is applied to Z74, pin 1Ø. It also is outputted by the decoder section as RAM*. RAM* will select the $\overline{CS}$ pin on all of the RAMs, after it passes through DIP shunt X72. (It's shown on sheet 2 of the large Schematic.)

The selection of the data bus for RAMs is handled the same way during a ROM-Read operation. MEM* will go low because RD* went low. But during a CPU data dump from CPU to RAM, MEM* does not select the data bus buffers for the RAM. Instead of RD* being active, WR* is low. We don't need the ROM/RAM buffers because the RAM data inputs are on the output side of the buffers. Only during a ROM/RAM read operation do we need MEM*.

Notice on X3 that we can program the system for 8K of RAM by leaving the shorting bar intact at pins 3 and 14, and at the 2 and 15 position. Not only would a 4000 address cause RAM*, but a 5000 address would also enable RAM*. If we had 12K of RAM, we would leave pins 4 and 13 shorted. For 16K, we short all pins we have mentioned; plus pins 5 and 12. RAM* would now be active from addresses 4000 to 7FFF.

As you can see from the RAM discussion, we'll be shorting certain outputs of Z21 together. In most applications using TTL, shorting output nodes is bad design practice. But there are some "open collector" type TTL devices. These types of gates do not have an active pull-up on the output. Instead, the output transistors have "open collectors". It is the responsibility of external circuitry to pull them up. The "open collector" outputs may be tied together for a "wire OR" function.

Since Z21 is an open collector decoder, the output may be safely tied together. Notice resistors R48, R61, R62 and R68. These are the pull-up resistors for Z21.

Something to remember about open collector outputs: You cannot tell if one of these outputs is working unless there is a pull-up resistor tied to that output. For example, if you placed an oscilloscope probe on pin 10 of Z21 as shown in Figure 3, you would not be able to tell if pin 10 goes low. If the system is working right, it shouldn't. But if it isn't working right and pin 10 is going low, how are you going to prove it? Pull it up with a resistor to +5 volts and see; that's the only way you can be sure.

System RAM

According to the Block Diagram, System RAM is tied in parallel with the data bus and address bus just like ROM and the Keyboard. The data input and output for RAM is straightforward enough; MEM* controls the buffers. But the addressing scheme appears all screwed up. How can the CPU address a minimum of 4K of RAM using only seven address inputs? The answer to that very good question is — multiplexing. The address from the CPU is multiplexed into the RAM in two 7-bit parts. The RAM's internal logic takes the two parts and brings them together to form one address scheme with 14 bits. One part of the addressing is called RAS* (ROW Address Select); the other part is CAS* (Column Address Select). Another signal, MUX (Multiplexer), controls the switching function. All three of these signals are generated near the CPU on Sheet 1 of the large Schematic.

MUX, CAS* and RAS*

On Sheet 1, find pins 21 and 22 of the CPU. Follow the lines tied to these two pins down to NAND gate Z74. If we get a low at $\overline{WR}$ (Memory Write) or a low at $\overline{RD}$ (Memory Read), Z74, pin 3 will output a high (called MREQ, Memory Request). MREQ is tied to the clear inputs of Z69 and part of Z70. These devices are D type flip-flops where the MUX *and CAS* signals are generated.

Figure 4 shows a waveform chart for this circuit. Line A shows the master clock input to the flip-flops. Line B shows $\overline{MREQ}$ and Line C depicts the $\overline{WR}$ output from the CPU. Assume that the CPU wants to write data into RAM. As shown on Line B, $\overline{MREQ}$ will go low. A short time later, $\overline{WR}$ will go low. Line D shows Z74, pin 3, going high at the same time $\overline{WR}$ went low. The flip-flops now have a logical high applied to the clear inputs.The flip-flops are free to operate, controlled by the clock waveform. On the next rising edge of the clock, Z69, pin 5 will output the logic level that was present at pin 2 the instant pin 3 went high. Since pin 2 was high when pin 3 went high, pin 5 will go high. This high is shown on Line E. Z69, pin 12 is now high; so on the next rising edge of the clock, pin 9 will go high. This is shown on Line F. Z70 is ready to toggle. On the next rising edge of the clock, Z70, pin 6 will go **low** (Q went high, so $\overline{Q}$ must go low). This is shown on Line H of Figure 4. All three flip-flops have changed states since $\overline{WR}$ went low. The flip-flops will stay in this state as long as $\overline{WR}$ stays low. When $\overline{WR}$ goes high, the flip-flops will have a low applied to their clear inputs; and they will reset back to the clear condition.

Line J is the RAS* output. As you can see, it is a direct function of $\overline{MREQ}$ from the CPU. Z72, pins 4 and 5 are RAS*'s buffer. Line K, MUX is sourced at Z69, pin 9, through buffer Z72, pins 2 and 3. Line L, CAS*, is buffered by Z72, pins 10 and 9, and is a function of QCAS, at Z70, pin 6.

Notice the following sequence of events: RAS* goes low first. MUX then changes states. CAS* then changes states one clock cycle later. First, we get ROW Address Select, then MUX, then we get Column Select. Thus, the first part of the address we give the RAMs will be the row address. We'll then flip the switch (multiplexer) and follow with the column address.

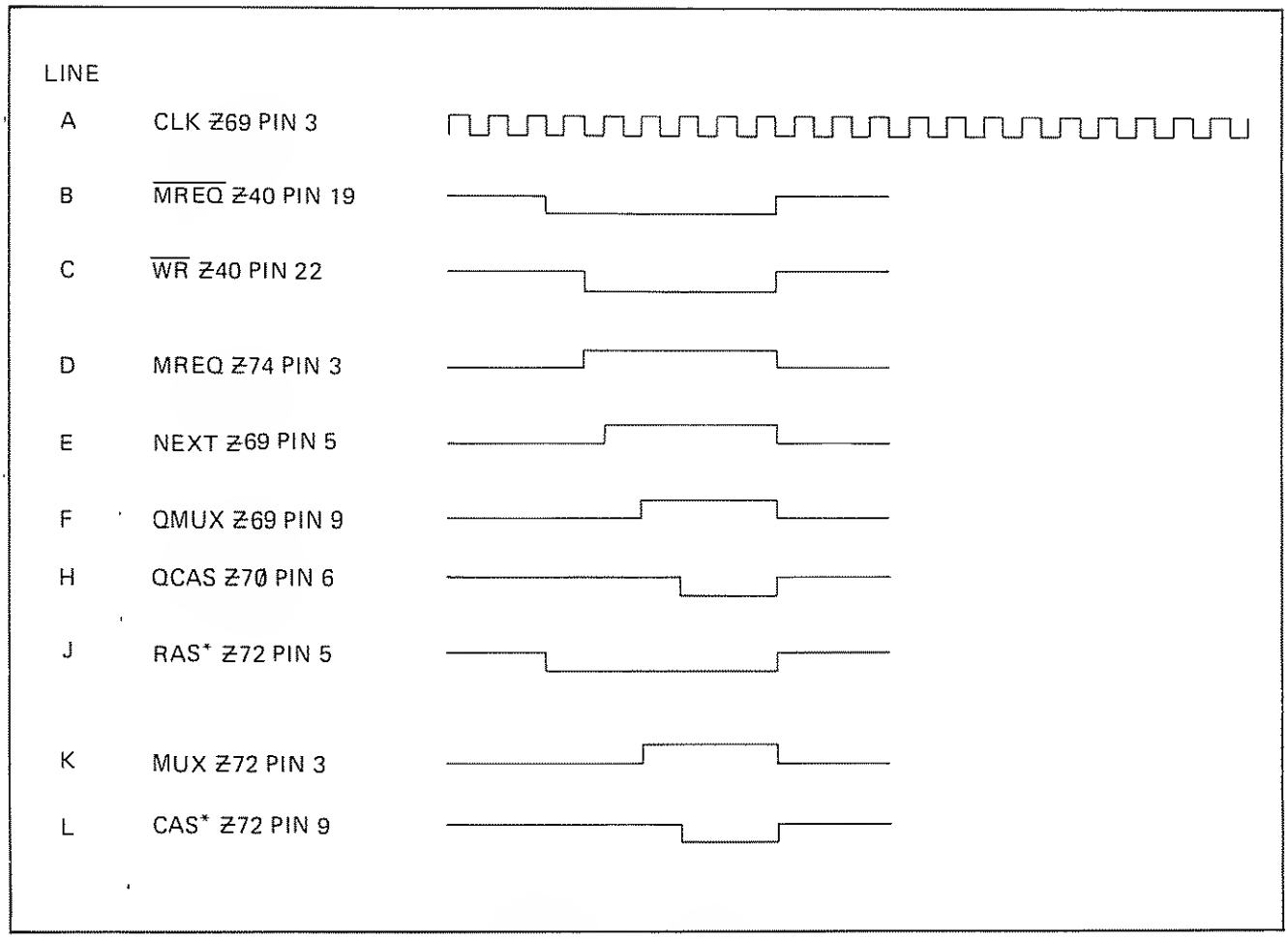


FIGURE 4. Waveform Chart

RAM Addressing

In about the middle of Sheet 1, on the left side of the RAM array, multiplexers Z35 and Z51 are shown. On the left side of Z35, we find the area where four address lines are coming in. One brace of four is labeled "Ø" and the other is labeled "1". Z51 is configured the same way except there are only three lines per brace. The Ø tells us that when the select pin is low, the multiplexer will be outputting data associated with these input lines. The "1" tells us the opposite is true. When the select pin on the multiplexer goes high, it will be outputting data associated with the "1" input address lines. The select input for both multiplexers is pin 1. Z35, therefore, operates somewhat like a 4-pole, double throw switch, where the select input (pin 1) is doing the switching. Z51 is used only as a 3-pole, double throw switch — one input/output is not used. The enable input to the multiplexer is pin 15. Since pin 15 is grounded on both IC's, the "switches" are always enabled.

Reading From RAM

Assume the CPU needs RAM data. Let's follow the addressing and data paths the RAM will use. We'll use a 4K RAM example.

The CPU outputs a $\overline{\text{MREQ}}$ and a $\overline{\text{RD}}$. The address decoder outputs MEM* and RAM*. MEM* activates the RAM/ROM data buffers and RAM* enables the chip select ($\overline{\text{CS}}$) for the RAMs. At the same time, the multiplexer will load the address into the RAMs. RAS* goes low. The MUX signal is low at this time, so AØ through A5 on the RAM receive the low order address. Notice that RAS* is buffered by Z68, pins 14 and 13, and is applied to pin 4 of all the RAMs. The negative going signal at pin 4 will load the lower order address in the row section of each RAM. A short time later, MUX changes state; it goes high. The multiplexer, Z35 and Z51, now switch and the high order addresses are applied to the RAMs. CAS* will now go low. CAS* is applied to buffer Z67, pin 14. Pin 13 of Z67 passes CAS* to pin 15 of all eight RAMs. On the negative transition of CAS*, the high order addresses (A6 through A11) will be loaded in the column section of each RAM. The RAMs now have the entire address from the CPU. The RAM will now output data through the buffers and to the CPU.

Writing To RAM

The difference between a write operation and a read operation is exactly two signals. Address decoding and address multiplexing work the same way. During a data write, however, the CPU sends data to the RAMs. Thus, the ROM/RAM buffers are not needed; and MEM* will not go low. Instead of the CPU issuing a $\overline{\text{RD}}$ command, it supplies a $\overline{\text{WR}}$ instruction. WR* is tied to all eight RAMs on pin 3. When this pin is low, data will be stored in RAM at the specified address. When this pin is high, the RAMs are in read cycle.

Refreshing the RAMs

The TRS-80 uses dynamic type RAM. A dynamic RAM differs slightly from a static RAM in data retention. A static RAM will retain data stored in it as long as power is applied to it. A dynamic RAM must be periodically addressed to ensure that it retains the data loaded into it. The periodic addressing is called "refreshing." You might compare a dynamic RAM with an air-filled tire with a slow leak. Every once in a while, the tire must be shot a little air so it won't go flat. If we don't service that tire, it would finally become unusable. The same is true of dynamic RAM. If the system does not access the RAMs every so often, they will "forget" data.

The dynamic RAM in the TRS-80 uses an "$\overline{\text{RAS}}$ only" type of refresh. In other words, when RAS* goes low, the RAMs in the system will "refresh themselves" even though the RAM may not be in use at the time. As stated before,

RAS* is generated by the CPU at pin 19 ($\overline{MREQ}$). Whenever $\overline{MREQ}$ goes low, RAS* goes low; and the RAMs will load the lower order address into the row section. The CPU may be looking at system ROM when $\overline{MREQ}$ goes low, but RAM will still receive RAS* and thus be "refreshed."

Normally, you would not be too concerned about this aspect of the RAMs. But you need to be aware of the differences between a static RAM and a dynamic RAM. Remember: Dynamic RAM must be periodically addressed to enable it to retain data. In the TRS-80, the RAM is refreshed once every two milliseconds.

RAM Programming

You may have noticed X71 during the discussion of the RAM. X71 is a DIP-shunt. It is used to program the size of memory in a system. Find pin 13 on the RAM. Following pin 13 down, you will see it is tied to two pins of DIP-shunt X71. Pin 13 of the RAM is the CE (Chip Enable) or the A6 address input. In a 4K system, pins 4 and 13 of X71 are shunted. RAM* is on pin 4 so RAM* is used to select RAMs. But in a 16K system, 4 and 13 are opened and pins 3 and 14 of Z71 are shorted. Instead of RAM*, we'll get address line A6 or A12 (depending on multiplexer status) going to pin 13 of the RAMs. There are other parts of X71 shown on the left side of the multiplexer, Z35 and Z51. Before troubleshooting a system, you will need to know the size of RAM the system uses. If X71 is "programmed" wrong, you may find yourself with RAM problems.

Video Divider Chain

The Video Divider Chain supplies the Video RAMs with addresses in a logical order for Video Processing. This chain also supplies the horizontal and vertical sync timing pulses so that the Video Processor can build the composite waveform for the display. Video RAM addresses, horizontal and vertical sync, and Video Processing timing are all direct functions of the master clock. Also included in the Divider Chain is the hardware necessary to generate 32-character line-lengths. Although Level I can not access the 32-character format, Level II can.

Divider Chain Input Conditioning

If the TRS-80 did not have to change character line formats, the Divider Chain could have been tied directly into the master clock. But, the TRS-80 does have two formats for character lengths. In the most familiar format, the display has 16 character-lines, each consisting of 64 characters. This means there are 1Ø24 character locations in Video RAM which the Divider Chain must access. In the other format, the characters appear twice as large. The display will show 16 character-lines of 32 characters. The Divider Chain must access only 512 Video RAM locations. Switching from one format to the other is the job of the Input Conditioning logic.

On Sheet 2 of the large Schematics, the master oscillator circuit is surrounded by a D flip-flop (Z7Ø), a divide-by-12 counter (Z58) and a multiplexer (Z43). The D flip-flop is wired to perform a divide-by-two function. The multiplexer is wired so that we can route the master clock frequency, or the clock frequency divided by 2, from the flip-flop to the divide-by-12 counter. Since there are two character length formats, there must logically be two reference frequencies: one is half as slow as the other. The master oscillator supplies the divide-by-12 counter with a reference frequency in a 64 character format. The D flip-flop supplies the counter with the reference frequency in a 32 character format.

The multiplexer is doing the selecting, so what is controlling it? Pin 1 of Z43 is a signal called MODESEL (Mode Select). When low, MODESEL forces Z43 to be switched into its 32 character position. When high, MODESEL forces Z43 to be switched into its 64 character position. Let's look at the 64 character mode first.

Since MODESEL is high, pin 3 is "shorted" to pin 4 of Z43. Pins 6 and 1Ø are "shorted" to pins 7 and 9. (Remember: a multiplexer is an electronic equivalent of a multipole, double throw switch.) Figure 5 is a waveform chart for this circuit. At Line A, the master clock is shown at the output of its buffer, Z42. Line B shows the action of D flip-flop during its divide-by-2 function. The buffered clock is applied to pin 3 of Z43. Since the multiplexer is switched into its "1" state, pins 3 and 4 are the same signal and counter Z58 receives the 1Ø MHz clock frequency at pin 14. Notice that flip-flop output Z7Ø, pin 9, is tied to pin 2 of Z43. It is not performing any function at this time since the multiplexer is not switched into its "Ø" state.

The output of Z58 is shown at lines C, D, E and F in Figure 5. The arrows in this figure point out the place where Z58's outputs are all zero. Notice that Lines C through F do not count up to 11, then go back to zero using straight binary. Z58 starts fine: Ø . . . . 1 . . . . 2 . . . . 3 . . . . 4 . . . . 5 . . . . On the next clock, it goes from binary 5 to binary 8. From 8, it counts normally to binary 13; then on the next cycle, it goes back to binary zero.

Notice pins 6 and 7 of Z58. These inputs are used to clear the counter to zero. If you find CTR on Sheet 1, you will see it comes from inverter Z42, pin 8, which controls the CPU CLK divider. Normally, CTR is held low. Only during automatic testing at the factory is CTR allowed to go high and clear Z58. You might find "A"

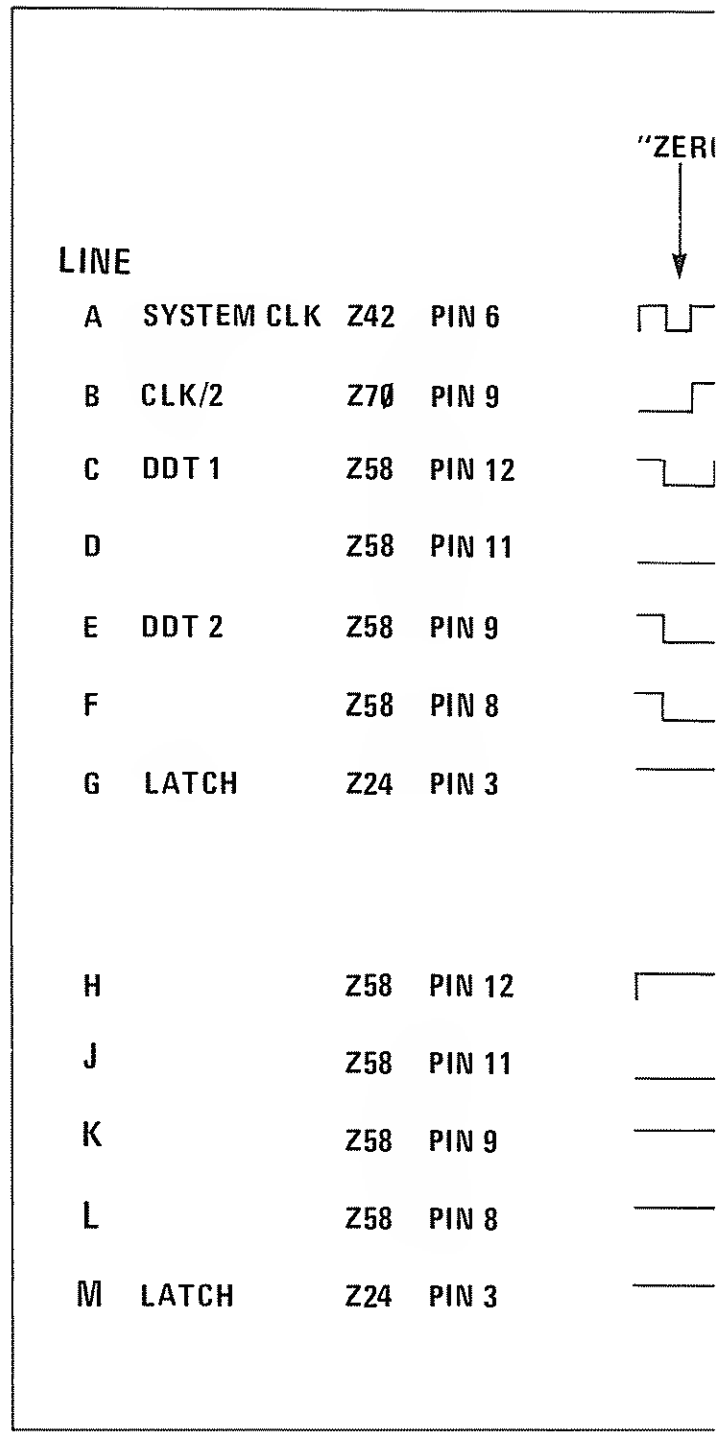
and "D" version P.C. Boards with Z58, pins 6 and 7 simply tied to ground.

Z58, pin 12, is labeled DOT 1. Z58, pin 9, is labeled DOT 2. DOTs 1 and 2 are "NAND-ed" by Z24; and the resulting output is shown in Figure 5 at line G. This signal is called "LATCH" and is used in video processing.

Z43, pins 6 and 10 are tied together and are connected to Z58, pin 8. The resulting output at Z43, pins 7 and 9, will therefore be the same signal. Pin 9, labeled CHAIN is the divider chain's main source. Pin 7 of Z43 is labeled "C1" and is tied to pin 10 of Z64, one of the Video RAM multiplexers. C1 will be used to address the Video RAM's least significant bit.

In the 32 character format, Z43, pin 1 will be low. Therefore, pins 2, 5 and 11 will be "shorted" to pins 4, 7 and 9 respectively. (The electronic switch was flipped.) Now we have the frequency source from Z70, pin 9, tied to counter Z58. Pin 7 of Z43 is held low all the time; and pin 9 of Z58 is now used as the source labeled CHAIN. In Figure 5, lines H through L show Z58's outputs.

Remember: We are using Line B in the Figure as the input to Z58 instead of Line A. Notice that Z58 is now being used as a divide-by-6 counter. The output at pin 9 is now CHAIN instead of pin 8. Has the CHAIN frequency changed? No. In 64 character mode, we had the master clock, divided by 12, as the chain frequency. That is, 10.6445 MHz divided by 12 = 887.041 kHz. In 32 character mode, we had 1/2 master clock divided by 6, as the chain. 10.6445 MHz divided by 6 divided by 2 = 887.041 kHz. What did change? Two signals changed. LATCH is sourced at Z24, pin 3. In 64 character format, the latch pulse was only one clock cycle (Master Clock) wide, having a period of 6 clock cycles. In 32 character mode, the pulsewidth has doubled to 2 clock cycles and its period is now 12 clock cycles. The other signal that changed



was C1. Sourced at Z43, pin 7, it was a square wave at the same rate as the chain signal; but in 32 character mode, it is held low all the time.

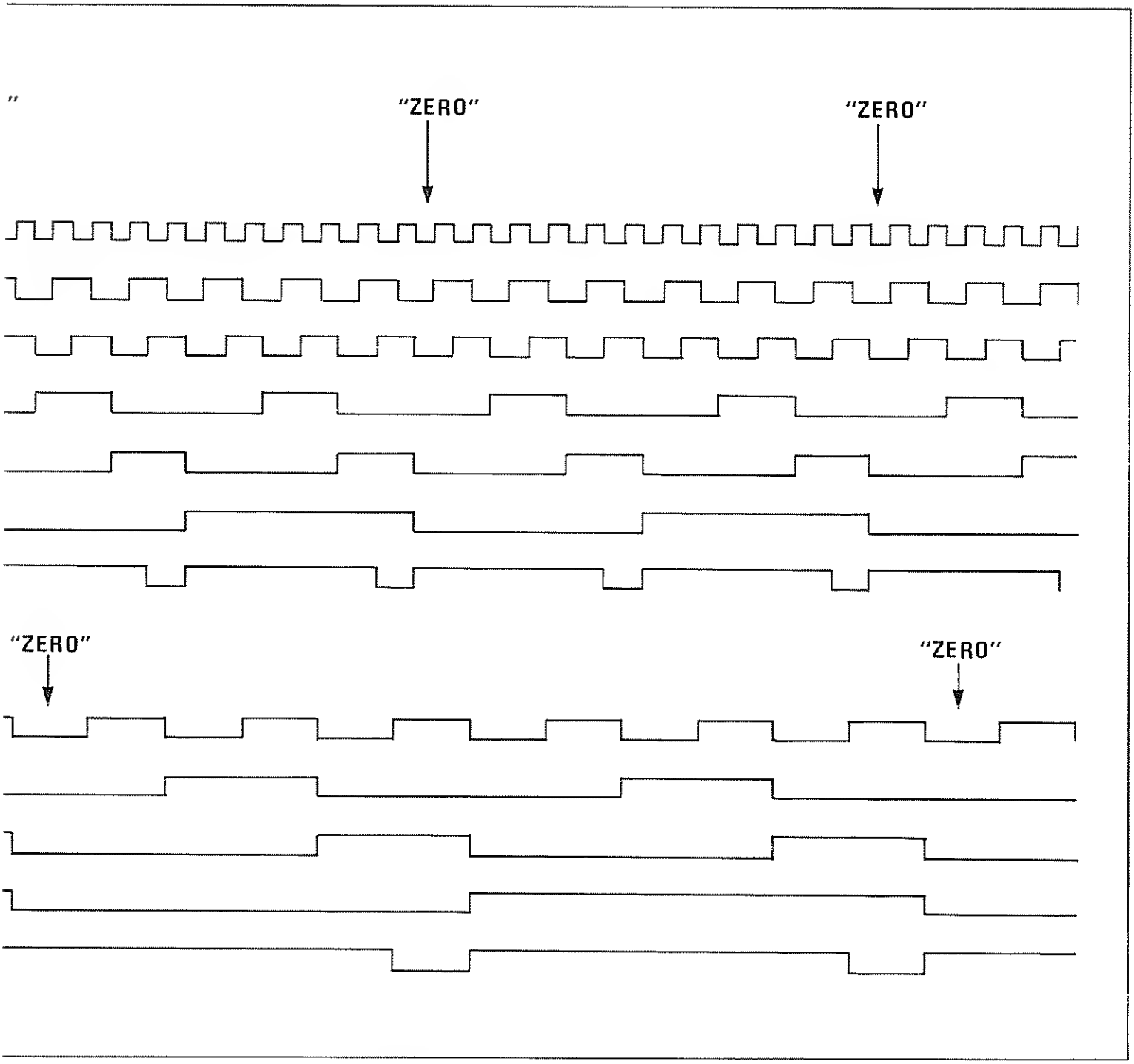


FIGURE 5. Input Conditioning Waveform

The signals that changed are very important to the Video Processor section. The first, LATCH, is used to delay a character between RAM and the character generator. The second signal, C1, determines if the RAM has 1Ø24 or 512 useable addresses.

Divider Chain

The Divider Chain circuit (Z65, Z5Ø, Z12 and Z32) consists of four-bit ripple counters. They hve a maximum count of 16, but external circuitry may modify this maximum.

Figure 6 shows a simplified block diagram of the counter chain. Refer to Figure 6 and to Sheet 2 during our discussion of the counter chain.

Z65 is a binary counter that is split into two parts. The chain input from the conditioning logic is applied to pin 1 of Z65. The B and C outputs are used for Video RAM addressing, and the output of Z65 at pin 8 is applied to the next counter in the chain. This part of Z65 divides the chain frequency by 4. Since the chain is 887.Ø461 kHz, the output of Z65, pin 8, is 221.76Ø kHz. The other part of Z65 will be used later.

The next counter in the chain is Z5Ø. The input is on pin 14 and the divided frequency is at pin 11. This device is externally modified to divide the input frequency by 14. Z5Ø counts up normally to a binary value of 13. Thus the counter's outputs are as follows:

Pin 12	(Output A)	=	1
Pin 9	(Output B)	=	Ø
Pin 8	(Output C)	=	1
Pin 11	(Output D)	=	1

Upon the next negative excursion of the clock pulse, outputs would look as follows:

Output A	=	Ø
Output B	=	1
Output C	=	1
Output D	=	1

which is equal to 14. But notice AND gate Z66, pins 3, 4 and 5. These pins are tied to outputs B, C and D. The output of the AND gate's pin 6 will go high and clear Z5Ø back to zero. This clear pulse is extremely rapid — about 5Ø nanoseconds! The binary count of 14 would therefore be almost invisible to a standard oscilloscope and so would the clear pulse to pins 2 and 3. The time that Z5Ø is actually reading binary 14 is so short that we can ignore it. Therefore, Z5Ø will count from Ø to 13 and will then reset back to Ø. Since 221.76Ø kHz is put into Z5Ø, the output at pin 11 will be 15.84Ø kHz. This frequency will be used by the sync generator circuits to produce horizontal sync.

The next divider is Z12. It is wired to perform a division by 12. It counts up normally until the outputs enable AND gate Z66, pins 9, 1Ø and 11. This happens at the twelfth falling edge of the clock. Z66, pin 8, will then go high and clear Z12 back to zero. Once again, this clear pulse would be very hard to observe using an oscilloscope. Thus we can ignore this count and con-

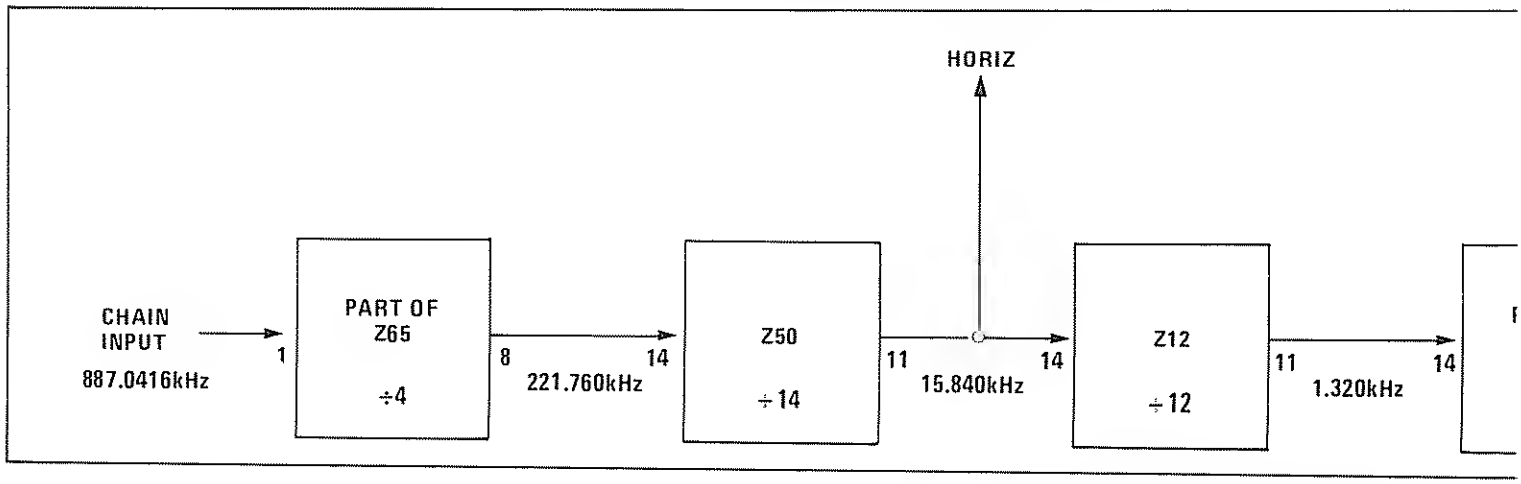


FIGURE 6. Divider Chain Block Diagram

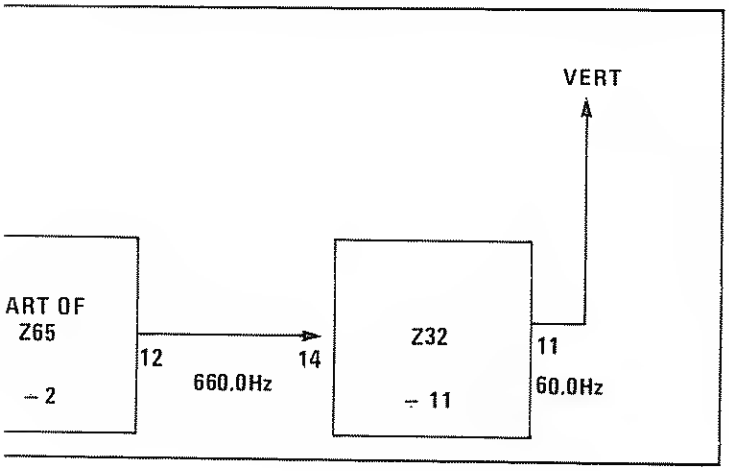
sider Z12 as a divide-by-12 counter instead of a divide by 13 counter! If 15.84Ø kHz is applied to Z12, pin 14, then the output at pin 11 will be 1.32 kHz.

The next divider is part of Z65. On Sheet 2, follow pin 11 of Z12 up to Z65, pin 14. The output is at pin 12. Follow it back down to Z32, pin 14. This part of Z65 divides the 1.32 kHz input by two; therefore, the frequency at pin 14 of Z32 will be 66Ø.Ø Hz.

Z32 is the last counter in the chain. It divides the 66Ø Hz input by 11, producing 6Ø Hz. Once again, part of Z66 is used to modify the count. When the outputs of Z32 equal binary 11, Z66 will output a very narrow pulse which clears Z32 back to zero. The 6Ø Hz at pin 11 is used by the sync generator circuits to produce the vertical sync for the monitor.

Video RAM Addressing

During our discussion of the System Block Diagram, you noticed that the Video RAMs must be addressed from two sections. The CPU must address Video RAMs to read or write data from or to specific locations. The Divider Chain must also address video RAM so that data contained in memory can be processed and displayed on the screen. The Video RAMs are addressed either by the CPU or by the Divider Chain through the use of three multiplexers.



Z64, Z49 and Z31 are the three multiplexers used for Video RAM addressing. From the Divider Chain, there are 1Ø address lines that will be used to address Video RAM. The chain conditioning logic supplies one address: C1. Z65 supplies three addresses: R1, C2 and C4. Z5Ø supplies three addresses: C8, C16 and C32. Z32 supplies the rest: R2, R4 and R8.

Imagine an array of rectangles; 16 rectangles vertically and 64 rectangles horizontally. You would have a total of 1Ø24 rectangles. You could specify any one rectangle by saying, "Starting at the top left hand corner, go down four rows and go to the right 18 columns." The 16 rows could be assigned a binary number from Ø to 15. The 64 columns could be assigned a binary number from Ø to 63. Rectangle Ø—Ø would be in the upper left hand corner of the array. Rectangle 15—63 would be in the lower right hand corner. Four bits of binary information would therefore specify any one of the 16 rows. It takes six bits of binary data to specify any one of the 64 columns. This is exactly the addressing format used by the counter chain. C1, C2, C4, C8, C16 and C32 specify any column. R1, R2, R4 and R8 specify a row. The row/column addressing format is very useful in troubleshooting video problems in the TRS-80.

The column and row address outputs from the Divider Chain are applied to the "1" inputs of the multiplexer. Part of the CPU's address bus is tied to the "Ø" input of the multiplexer. The outputs of the multiplexer are tied to the Video RAMs or logic around them.

We've got inputs and outputs; how about control? Do you remember the signal VID* that we

generated back in the address decoding discussion? We said VID* will select the Video RAMs. Notice that pin 1 of the 3 multiplexers is tied to VID*. When the CPU wants control over the Video RAM, the address decoder recognizes the Video RAM address and causes VID* to go low. When VID* is low, the multiplexer switches the "Ø" inputs over to the multiplexer outputs. The counter chain addresses are switched out of the circuit, and the CPU has control over Video RAM. When VID* goes back high, the CPU is switched out and the counter chain takes over.

Most of the time, the counter chain is in control of Video RAM. The CPU only takes charge when it needs to. You can see on the display screen when the CPU robs the counter chain of Video RAM control. Ever notice black streaks all over the screen while graphics are being drawn? These streaks are the result of the counter chain losing control over Video RAM.

Aside from chain and CPU address, there are inputs to the multiplexer we have not yet mentioned. The first of these inputs is the resistor at pins 13 and 6 of Z49. These two inputs, which are not needed in the counter chain's control over Video RAM, are pulled up to 5 volts by R49. Outputs pins 12 and 7 correspond to the inputs at pins 13 and 6. When the chain has control over Video RAM, pins 12 and 7 output a steady state high. Pin 12 goes to the R/$\overline{W}$ (Read/$\overline{Write}$) control of all the RAMs. Since the counter chain never stores data in RAM at the address it specifies, pin 12 should be high when the chain is in control. Pin 7 of Z49 goes to the Video RAM data buffer. When the chain is in control, the RAM data bus should be disabled. A high on VRD* (Video Read) guarantees this bus will be off.

We also find WR* and RD* tied to pins 14 and 5 of Z49. When the CPU takes charge of the Video RAMs, multiplexer output at pin 12 becomes VWR* (Video Write). The CPU can store data into the video RAMs by causing VWR* to go low. If the CPU wants to read data from video RAMs, RD* can pass through Z49 and activate VRD*. A low here will open data buffers Z6Ø and Z44. Addressed Video RAM data is then placed on the data bus. The CPU can process this data like any other data.

Alphanumeric Format

The CRT (Cathode Ray Tube) in the display will be scanned twice per second. The electronic beam in the CRT travels from top to bottom of the screen and left to right. Each screen or frame consists of 264 scan lines. 192 scan lines are used in the "picture". 72 lines are used during vertical interval and as upper and lower boundaries. Nothing is ever "written" or visible within these 72 lines. There are 1Ø24 character locations per screen (or 512, depending on status of MODESEL). Each character-line consists of 64 characters (or 32, depending on the status of MODESEL). There are 16 character lines. Each character-line consists of 12 scan lines. An alphanumeric character uses seven scan lines while there are five blank scan lines between character lines. We'll worry about graphics formatting later.

Part of Z65 and Z5Ø specify the column address. Z32 specifies the row (or character-line). Z12 specifies the scan line in any character line. The outputs from Z12 are labeled L1, L2, L4 and L8. These four lines are not used in Video RAM addressing because we already stated a row and column address will specify any one of the 1Ø24 rectangles in our rectangle array. Z12's outputs are used in the Video Processing. L1, L2 and L4 will enable the character generator to output correct data for any character since it knows where the CRT's electron beam is scanning. L8 is used by the Video Processor to blank (turn off) the five lines between character lines.

Notice NOR gate Z3Ø, pins 8 and 9. These pins enable Z3Ø's output to produce a signal called BLANK*. BLANK* is used by the Video Processor to give the 72 scan line blanking for the upper and lower boundaries. It also defines the blank boundaries on the left and right of the screen.

Video RAMs

The Video RAMs are static and do not need refreshing. The data bus is wired in the same way as system RAMs, but with a different enable signal.

One interesting point to note: There are seven RAMs. Six are used for ASCII storage, and the seventh is used as a graphic/alphanumeric definition bit. There are eight data lines. Notice the line labeled Bit 6. It is sourced by NOR gate Z3Ø at pin 13. If Bit 5 (Z62, pin 12) and Bit 7 (Z63, pin 12) are low, then Bit 6 will be high. Z3Ø is a sneaky way of squeezing a seventh ASCII bit out of six RAMs.

Aside from the data bus, there is another RAM output for data. The Video Processor needs video data for generation of the alphanumeric and graphic symbols. And this section will be discussed next.

Video Processing

Video Processing consists of five subsections. They are: Data Latch, Character/Graphic Generator, Shift Register, Sync Generator, and Video Mixing/Output driver.

The Data Latch temporarily stores an ASCII or graphic word from Video RAM. The Latch will retain the byte for processing so that the RAM is free to search out the next byte. **The Character Generator** is a ROM that is addressed by the Data Latch and the scan line signals. This ROM contains the alphanumeric format that makes up all the characters. **The Graphic Generator** is not a ROM, but a 4-line-to-1-line data multiplexer. It operates somewhat like a "bit steering" circuit. It steers an ASCII word into a graphics symbol. **The Shift Register** accepts data from the character generator (or the graphics generator) and converts parallel dot data into serial dot data. Meanwhile, the **Sync Generator** circuits have been accepting timing signals from the divider chain. The Sync Circuits shape up the horizontal and vertical pulses, serrate the vertical interval and send it all out to video mixing in serial format. In the **Video Mixing** section, the serial dot video and the serial sync are brought together. The resulting composite video signal is then "fine tuned" in amplitude and dot-to-sync ratio, and then buffered for a 75 ohm output cable. The signal leaves the TRS-80 and is applied to the Video Display. In the Display, the signal is torn apart into its separate components; and you have a readable image on the screen.

Data Latch

The Data Latch consists of Z2B for the ASCII and Z27 for the graphic bit and blanking signals. The inputs of Z28 come from the six Video RAMs. The outputs of Z28 go to Z29, the alphanumeric Character Generator and to Z8, the Graphics Generator. The inputs which control Z2B are on pin 9 (latch) and pin 1 (VCLR*). The latch signal at pin 9 is a pulse train developed by the divider-chain-input-conditioning logic. This signal goes low every six dot cycles (see Figure 5 for latch timing). On the rising edge of latch (low to high transaction), ASCII data in RAM is transferred to the outputs and temporarily stored by Z28. RAM data at the input to Z28 may now change, and the RAM has time to search for the next ASCII character.

Note: RAM, any RAM, has a parameter called "Access time". This is the time it takes for the data output to reflect a change after an address change. For example, assume a RAM output is high. The address of the RAM is changed to a new location where a "Ø" data bit is stored. Even though the RAM is now addressing the low cell, the output still reads the previous high. Only after a short length of time (in the nanoseconds) will the output change from a high to a low.

At the same time Z28 stored the ASCII word, the Divider Chain changed Video RAM addresses. The RAM is now "looking" for the next ASCII word. It has exactly six dot times (about 560 nanoseconds in 64 character format) to find it before the Latch is commanded to store the next word.

Z27 is a smaller latch that operates exactly the same as Z28. But instead of ASCII, it handles the graphic bit and blanking data. Pin 4 is tied to the inverted output bit from Z63, the graphic RAM. Pin 5 of Z27 has signal 8LANK* tied to it. L8 is tied to pin 12 and our "sneaky" bit, bit 6, is tied to pin 13 of Z27. All of these signals are latched into Z27 at the same time as the ASCII word is latched into Z28.

Each input to Z27 has a different function. The graphic bit to pin 4 of Z27 will determine if the ASCII word contained in Z28 is an alphanumeric character or is a graphic word. Pin 5 of Z27, is signal 8LANK*. This signal comes from Z3Ø, pin 1Ø, and controls the upper, lower, left and right boundaries of the video display. When 8LANK* is high, the CRT's electronic beam is allowed to draw on the screen. When 8LANK* is low, the beam is in a boundary area so it prevents the beam from drawing anything. L8 is connected to pin 12 of Z27. L8 acts somewhat like BLANK*. L8, remember, specifies where the electron beam is located in any **character-line**. When low, L8 allows the beam to output alphanumeric dot data. When high, L8 shuts off the beam because it is now scanning one of the five scan lines between character lines. The last piece of data comes into Z27 at pin 13. Sourced at Z3Ø, pin 13, this is the "sneaky bit" that is derived from data contained in RAMs Z63 and Z62. This is the only bit at Z27 which could be considered part of the ASCII word. The output is applied to Character Generator Z29, at pin 1.

Notice pin 1 of both data latches. This input, when low, will force the latches to their clear state (zero at the outputs). This signal is called VCLR* (Video Clear) and is sourced by D flip-flop Z7, at pin 6. The flip-flop disables the data latches during a CPU interruption of video RAM. Notice pin 4 of Z7. It is tied to VID*. When VID* goes low, Z7, pin 6 will go low. The low at pin 6 will clear the data latches. (This is what generates the black streaks we discussed in the Video RAM Addressing section.) When the CPU has finished with Video RAM, pin 4 of Z7 goes back high. The next time data is to be latched into Z27 and Z28, Z7 will toggle back to its normal reset state and allow the data latches to operate. If Z7 was not used, we might see characters that appear ripped apart on the screen. For example, assume the CRT was drawing a character when the CPU took command of Video RAM. After the CPU finished, the Video Processing circuit may still see the ASCII code that was in the latch at the time the CPU suddenly jumped in. The video circuit would try to redraw the character on the screen. We would then either see the character twice; or half of it would be over there, and the other half would be here! Clearing out the data latch insures that the Video Processor does not get confused.

Character Generator

Each character consists of a dot matrix. The matrix is five dots wide by seven dots deep. There is one dot between any two adjacent characters that is never turned on. We have five dots, a space, five more dots, a space, etc. Vertical spacing between adjacent data is determined by the frequency of the dot clock. (In the TRS-80, the dot clock signal is labeled SHIFT.) The dot clock is oscillator frequency, in 64 character format, and 1/2 oscillator frequency, in 32 character format. Horizontal spacing between adjacent dots is a function of scan frequency. In other words, each row of dots is aligned along the electron beam's path across the CRT. There are seven rows of character dots and five rows of blanks.

Since each character consists of a pattern of dots, there must be some method to determine which dot should be on and which dot should be off to form any one character. The character generator controls the dot patterns on the screen.

Z29 is the Character Generator. The seven bit ASCII word, stored in the Data Latch, is applied to Z29's ASCII inputs, pins 1 through 7. The ASCII addresses a certain area in Z29. You might consider the ASCII inputs to be the higher seven bits of an address. The lower part of the address is inputted at pins 8, 10 and 11. This three bit input selects the row position of the addressed dot pattern. Z29 outputs five dots at one time. Since each character consists of seven rows of five dots, the character generator must output seven separate times just to build one character.

Here is how a typical character line is written: Assume an ASCII word is in the Latch. The electron beam is on the first scan line of the character. Hence, pins 8, 10 and 11 have a binary "0" applied to them. Z29 outputs the first dot pattern for that particular ASCII character. The next ASCII character is applied to Z29. It outputs the first five dots for that character. This process goes on until the beam has scanned the entire width of the screen. If we could stop action at this point, all you would have would be a line of dots. On the second scan line, the data at pins 8, 10 and 11 is incremented to read binary "1" (001). The RAM is now prepared to read the second row of dots. The first ASCII character is applied, and it will output the second row of dots for that character. The second ASCII word comes in, and the second row of dots go out. This process continues until all 64 characters have had the second row outputted under the first row of dots. The line counter increments and we apply the first ASCII word once more. We paint a row of dots, increment the line counter and paint another row. Any character in a line is accessed at least seven times. Once the line counter has gone past the seventh count, all the dots make sense; and we will recognize the dot patterns as characters. After the seven dot scans are outputted, the electron beam is turned off; and five rows of blank dots are outputted. We would now be ready to output the first row of dot patterns for the second character line.

The dot output appears slow-reading about it. But ASCII is being shot into the character generator at about a 1.77 MHz rate. The CRT and the retention of the eye make these characters seem like they are outputted whole.

Graphics Generator

Do you remember the rectangle array we discussed in the Video Divider Chain section? Well, we are back to the rectangles. As stated earlier, there are 1024 character locations in Video RAM. If we divide each large rectangle into six smaller rectangles, we will have the basic graphics cell (Figure 7 shows a divided rectangle). This cell is the smallest piece of graphic information that can be displayed on the screen. Each cell is four scan lines long and three "dots" wide.

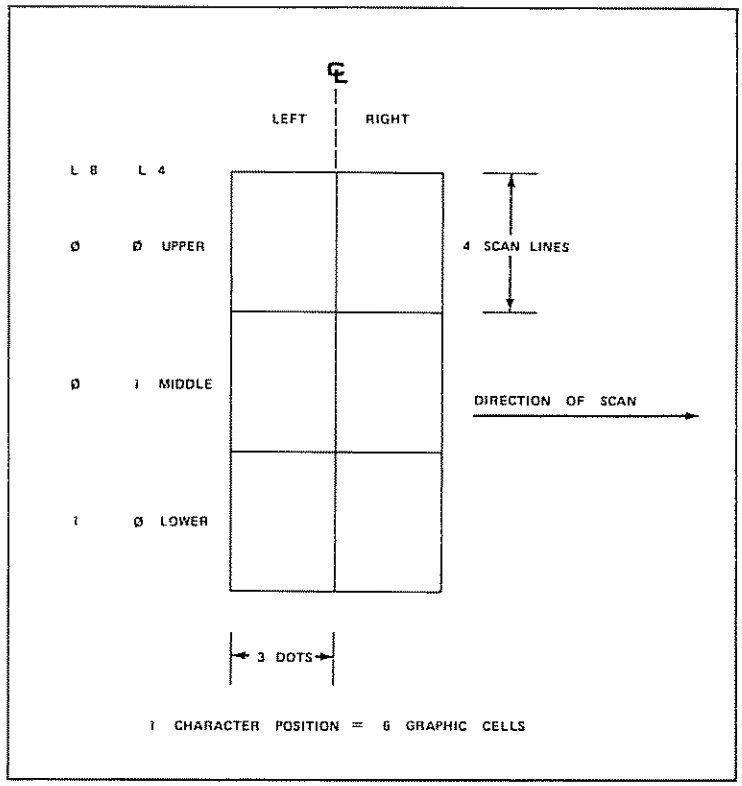


FIGURE 7. Rectangle and Graphic Cells

Z8 is the Graphics Generator. Actually, Z8 does not generate anything. Rather, it steers the ASCII addresses around to simulate a graphics generator. The input to Z8 is ASCII from Data Latch Z28, and the higher order line address from Z12, L4 and L8. L4 and L8 can represent any four numbers from Ø to 3. But since Z12 never goes to binary 12 (except for so short a time we can ignore it), we will only be looking for a binary number from Ø to 3. L8 and L4 are used to specify the vertical address of the six graphic cells. There are three vertical addresses: ØØ defines the uppermost pair of cells, Ø1 defines the middle pair of cells, 1Ø defines the lower pair. This is also shown in Figure 7.

The ASCII word, labeled L8Ø through LB5, determines if the graphics cell is ON (high) or OFF (low). The position of one of these inputs to Z8 determines which side of the center line the cell is located. An input at pin 6 of Z8 specifies a left hand graphic cell. Input at pin 1Ø of Z8 specifies a right hand graphics cell. Pin 5, left: pin 11, right; pin 4, left; pin 12, right. For example: Assume L82 is high and all other L8 inputs to Z8 are low. L82 comes into Z8 at pin 5. This pin is associated with a graphic cell location on the left of the character position. Therefore, depending on the status of L8 and L4, L82 will turn on (light) one of the graphic cells on the left of center line. If L3 and L4 are at logical ØØ, the upper left cell is turned on. If L3 and L4 are Ø1, the middle left cell will light.

As you can see, Z8's function as a graphics generator is to steer the ASCII bits around the character rectangle. The vertical position of the graphics in the cell is determined by the status of L8 and L4. The two outputs from Z8 are labeled "left" and "right". This "dot" information is applied to the Graphics Shift Register. It is in shift register logic that data from RAM Z63 determines if graphics or alphanumerics will be written in any one character position.

Alphanumeric/Graphic Shift Register

Z1Ø is the Alphanumeric Shift Register and Z11 is the Graphic Shift Register. Both devices receive parallel data from their respective generators. The parallel dot data is loaded into the registers and the dot clock (labeled SHIFT) will march the dots out, one behind the other, to the video mixer. We will discuss the Alphanumeric Shift Register first.

There are a few restrictions when the Alphanumeric Shift Register should serialize dot data and when it shouldn't. First, the data must be alphanumeric and not graphic. Second, the electron beam must be on one of the seven scan lines that are reserved for dot data and **not** on one of the five lines that are blanked (held off) between character lines. Third, the electron beam must be on one of the 192 scan lines that define the video portion of the screen. (Not in boundary space: upper, lower, left or right.) Once all three restrictions are met, the dot data is parallel-loaded into the Register. NAND gate Z26 insures all conditions are met before data is stored in Z1Ø.

Delay bit 7* is sourced from latch Z27, pin 2, and applied to pin 1Ø of Z26. When this input is high, data in Z63 is low, which defines an alphanumeric character.

Delay L8 is sourced at Z27, pin 11, and is connected to Z26, pin 12. When this input is high, the beam is scanning in a character line and not between character lines.

Delay blank is sourced at pin 7 of latch Z27 and is tied to pin 9 of Z26. When this input is high, the electron beam is in the video portion of the screen and is not located near a sync pulse or in some boundary region.

All three restrictions have then been met. Pin 13 of Z26 is tied to the inverted LATCH signal. When pin 13 of Z26 goes high, the dot load process will be activated by a low on Z26, pin 8. Upon the next clock pulse at pin 7 of Z1Ø, dot data will be loaded into the shift register. After LATCH goes back high (one dot time after going low), the Shift Register starts clocking dot data out at pin 13 in a serial stream (when LATCH goes high, pin 13 of Z26 goes low causing pin 8 to go high). Each time LATCH goes high, it forces ASCII and conditional data to be stored in Z27 and Z28. During this time, Z1Ø will **not** be shifting dots out at pin 13. Z1Ø only shifts data out when pin 15 is high. When low, pin 15 forces Z1Ø to load data from the Character Generator.

Notice the unused inputs to Z1Ø. Pin 9, the clear input, is pulled up, via R4Ø, to V_{CC}. When this pin is low (due to a short), you would have a blank screen. Pins 14, 3, 2, 1 and 6 are tied to ground. Pin 14 gives you that blank dot between adjacent characters. Pins 2 and 3 are not used, but are register inputs like pin 4 or 5. Pin 1 is for serial data input and pin 6 inhibits the clock input. They are not necessary in this application.

The Graphics Shift Register is Z11. Operation is almost the same as Z1Ø, except for the condition that must be met for use. The graphics conditions are as follows: First, Z63 must specify a graphics character instead of an alphanumeric character. Second, the electron beam must be in the video region of the screen. There are only two conditions restricting graphics. Since a character rectangle ends where another starts, there is no inter-character line blanking. If you turn on all of the graphic cells, you would have a full, large square with no holes and boundaries surrounding the square. Once all of the restrictions are met, graphic dot data may be loaded into Z11 for shifting to the video mixer. The other NAND gate in Z26 is used as the graphics load enable.

The inverse of delay bit 7* (DLY bit 7) is sourced at Z27, pin 3. It is applied to Z26, pin 4. When high, this input tells Z26 that Z63 does indeed contain a "1" which defines a graphic code.

Delay blank (DLY BLANK) is tied to Z26, pins 1 and 2. When high, this input tells Z26 that the electron beam is indeed in the video portion of the screen.

Once all conditions are met and LATCH goes low, Z26 will go low. Just like Z1Ø, Z11 will load dot data; and when pin 15 goes back high, the shift process will start. The six graphic dots are shifted out of pin 13.

Notice pin 9 of Z11 is pulled up by R4Ø. Likewise, pins 3, 2, 1 and 6 are tied to ground. But pin 14 is used this time. In graphics, there is not a blank dot space between character rectangles.

Sync Generator

The Sync Generator circuit accepts timing signals from the Divider Chain to develop horizontal and vertical sync pulses for the display. These pulses are used by the display to control the CRT's electron beam.

The sync pulses are generated by logic which operate like linear elements.

Z6, a CMOS inverter, is used to generate the horizontal pulse; and Z57 is used to generate the vertical pulse. The HDRV (Horizontal Drive) signal is sourced from the Divider Chain at Z5Ø, pin 11. This signal is buffered by Z6, pins 13, 12, 1 and 2, and applied to potentiometer R2Ø. R2Ø controls where the horizontal pulse starts in reference to HDRV. When R2Ø's wiper is close to Z6, pin 2, the horizontal pulse will start almost at the same time as HDRV goes high. When the wiper is moved in the opposite direction, there is a delay between the time HDRV goes high and the time the horizontal pulse starts. R2Ø is not performing this phase shift by itself. C2Ø, together with two inverters in Z6, form the complete shift network.

Here's how it works: HDRV goes high, causing Z6, pin 2 to go high (in this case about 5.Ø volts). A current flows through R2Ø charging C2Ø. While C2Ø charges, the voltage at pin 3 of Z6 slowly increases from zero as the current through R2Ø decreases. After a length of time, the voltage at pin 3 of Z6 will be high enough for pin 3 to "see" a high. Z6, pin 4 goes low, causing pin 6 to go high. C2Ø rapidly charges. Everything stays in this mode until HDRV goes low. At this point, C2Ø starts to discharge at the same rate it charged. When the voltage at Z6, pin 3 decreases to a logical "Ø" level, pin 4 will go high, causing pin 6 to go low. C2Ø rapidly discharges. The process cycle is now complete until the next time HDRV goes high. The time the voltage level at pin 3 of Z6 stays above the minimum logical "1" level determines the amount of shift from HDRV. The effect of R2Ø's position (which adjusts the delay time) on the screen is a horizontal shift of the video display.

After the horizontal signal is phase shifted, the horizontal pulse must be shaped. C21 and R43 form a differentiation network which creates a smaller pulse of known width from the shifted HDRV signal. Operation is quite simple. When Z6, pin 6 goes high, C21 and R43 differentiate the rising edge. A narrow pulse is passed to Z6, pin 11, inverted by pin 1Ø and inverted and buffered by Z6, pins 9 and 8. The net result is a pulse about four microseconds long, called horizontal sync.

The vertical sync phase shift operates in the exact same manner as the horizontal. Z57 is used as the active element, for which R21 and C26 form the delay network. The differential network consists of C27 and R44. Notice the only difference between horizontal and vertical circuits is the value of the two capacitors.

Horizontal and Vertical Mixing

Once the two sync pulses are phase-shifted and pulse-shaped, NAND gate Z5 is used to mix the two signals together and serrate the vertical interval. Figure 8 shows idealized waveforms around Z5.

At Line A, the horizontal pulses are shown. The source for this output is Z6, pin 8. At Line B, the vertical pulse is shown coming from Z57, pin 8. Z5, pins 1 and 2 are tied to the waveforms shown at Lines A and B, and the resulting NANDed output is shown on Line C. Line C in Figure 8 is now used as a source to NAND the horizontal and vertical syncs once more. Line D shows the result of NANDing Line C with Line A. Line E shows the result of NANDing Line C with Line B. Lines D and E are NANDed by Z5, pins 10 and 9. The resulting mixed sync waveshape is shown on Line F.

Notice two things about Z5. First, pin 8's output (shown in Figure 8, Line F) is "false" composite sync. In other words, it is inverted away from true form. Secondly, notice Z5 may be evaluated using Boolean algebra into a 2-input exclusive OR gate. The output at Line C may be expressed as $V\overline{H} + H\overline{V}$, where V is vertical sync at Line B and H is horizontal sync at Line A in Figure 8.

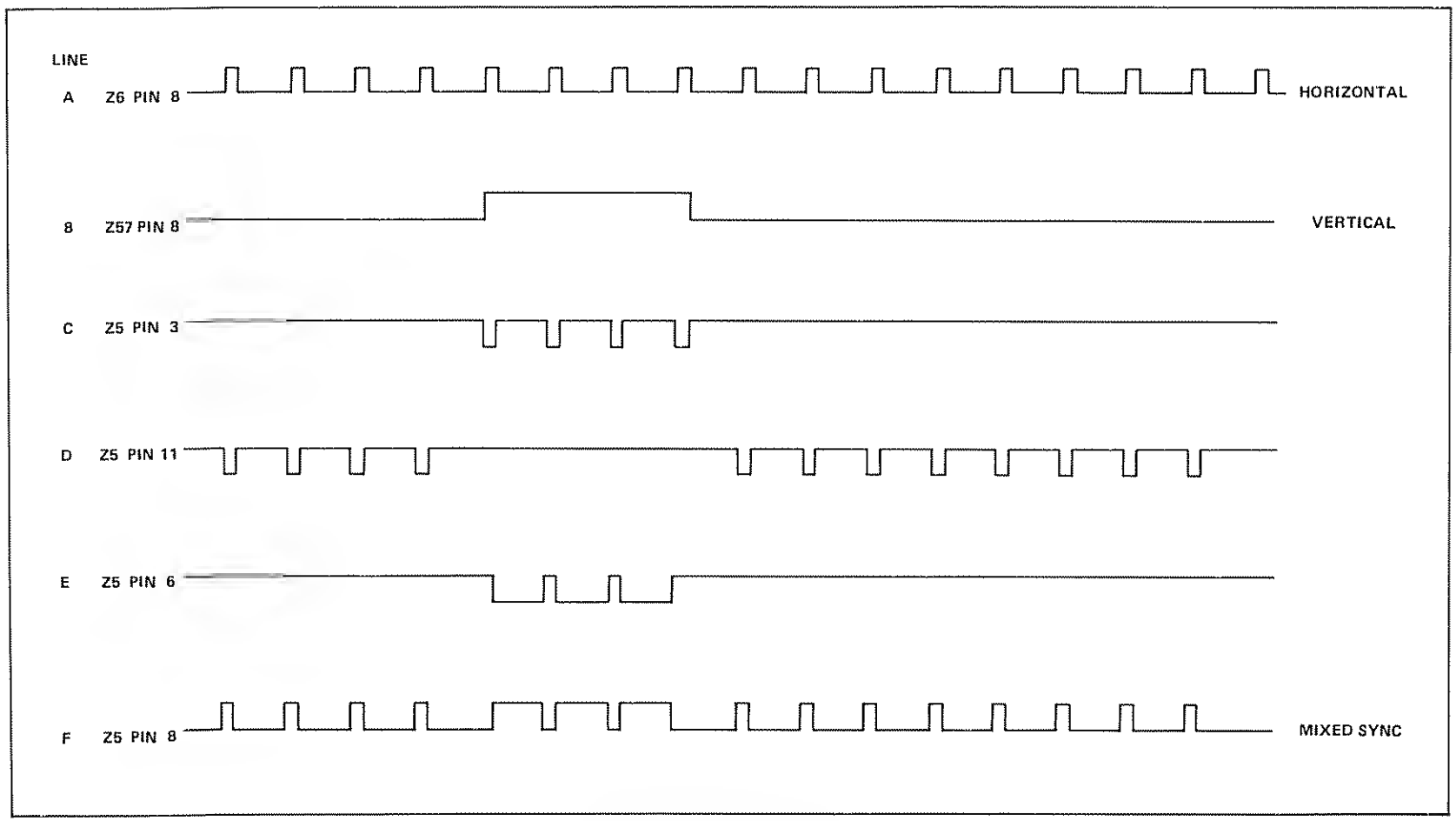


FIGURE 8. Sync Mixing

Video Mixing

The Video Mixing circuitry generates the composite video signal for the display. The video mixer accepts alphanumeric or graphics dot data from the shift register, level-shifts it, and places it atop the composite syncs. The composite waveform is then buffered to drive a 75 ohm impedance and is sent, via cable, to the Video Display.

Dot data from Shift Register Z1Ø or Z11 is applied to Z3Ø, pin 3 or pin 2. You should never see both pin 3 and pin 2 active at the same time. While Z1Ø is outputting alphanumeric data, Z11, pin 13 should be low. Conversely, if Z11 is outputting graphic data, Z1Ø, pin 13 should be low. The net result at pin 1, Z3Ø, is a single waveshape of video dot data. This data is applied to Z41, pins 6 and 7.

The composite sync data is sourced from Z5, pin 8, and is applied to the base of transistor Q2. Each time the base of Q2 goes to about Ø.6 volts below 5 volts, Q2 turns on, which applies 5 volts to resistor R28. (Actually, the voltage applied to R28 will be less than 5.Ø volts, due to the saturation voltage of Q2.)

The dot data from Z3Ø, pin 1, is inverted by Z41, pins 6 and 7. The resulting output at pin 5 is a normally-low signal which goes high only when the Shift Registers output a dot. Z41 is a high current driver. The output at pin 5 is the collector of the output buffer transistors. So, essentially, we have the video and sync going to two transistors. These transistors act as switches controlling current flow through resistor network R28, R27 and R23. Figure 9A shows a simplified drawing of the circuit.

Q2 and Z41 are represented as switches. When Q2 is "opened", there is no voltage applied to R28; and the output node is at ground level. When Q2 "closes" and with Z41 also held "closed", the output voltage goes up to about 1.23

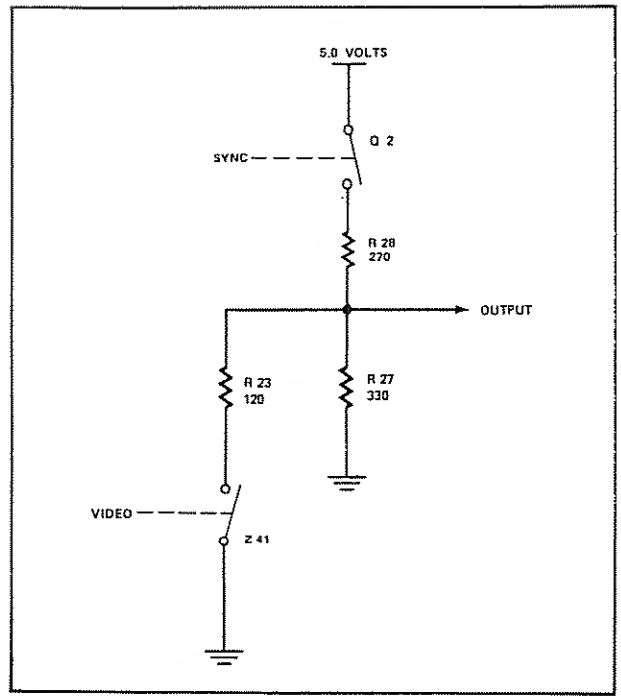


FIGURE 9A. Simplified Video Mixing

volts. This voltage will be called "the black level". Voltage below this level is "blacker-than-black" and is known as sync level. Voltage above 1.23 volts can be called the "white level". Normally, the black level stays at 1.23 until the sync at Z5, pin 8, goes high, turning off Q2, forcing the output at the node to go to ground. When dot data causes switch Z41 to open, the voltage at the output node increases to about 2.75 volts.

We now have a signal at the output node that contains both video dots and sync information. This signal is almost ready for the display. All that is necessary at this point is a little level shifting and output buffering.

Transistor Q1 is used as a common emitter amplifier. Composite video is applied to the base of Q1; and the emitter outputs the waveform shown in Figure 9B. This final signal is used by the Video Display. Capacitors C7 and C2, together with R3Ø, form a filter network for Q1's collector. The capacitors ensure the DC bias level on the collector is video-free, and helps in reducing power dissipation in Q1.

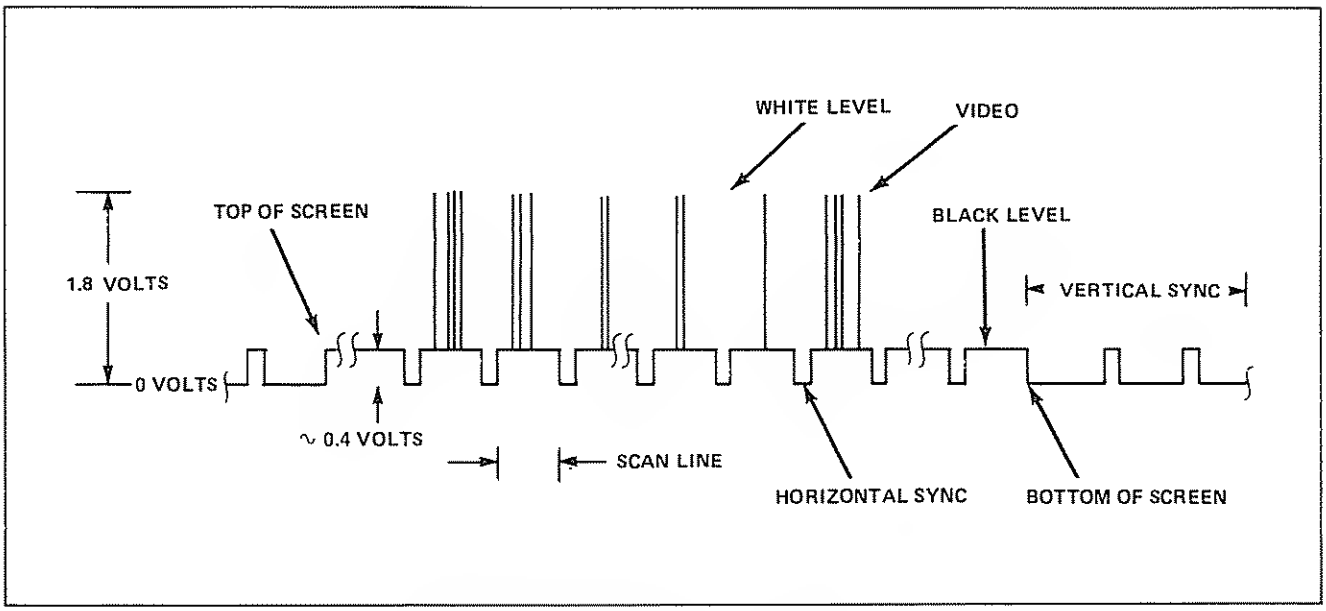


FIGURE 9B. Composite Video Output

Keyboard

The TRS-80 Keyboard consists of 53 single-pole, single-throw normally open keys molded in a plastic base. The base is mounted, together with four ICs and associated resistors, to the keyboard PC8. As you can see from the Schematic, this Keyboard does not output ASCII. It is scanned, like calculator-type keyboards. Each key represents a switch across a matrix node. When closed, the switch will short out a horizontal line to a vertical line. ROM software will detect the node short and generate ASCII equivalents for that particular key.

The Keyboard is accessed by decoder signal KYBD*. When this signal goes low, it enables tristate buffers Z3 and Z4. The inputs to these buffers are normally held high by the pull-up resistors (at the top of the keyboard schematic) R1 through R8. All of the horizontal address lines are made to go high at the same time as KY8D* goes low. If the CPU detects a logical "1" on one of the data lines, there is a key pressed on the Keyboard. The CPU ROM will then scan the address lines one-by-one until it finds the "1" output on the data bus again. After finding it, the ROM can instruct the CPU how to generate the ASCII code for that particular key. At this time, the CPU also checks the status of the two shift keys. If neither of these keys is pressed, the ASCII code is not modified. If a shift key is pressed, the ASCII is modified accordingly.

Only one point should be brought up about the Keyboard. The inverters on the address lines are open-collector types. You may not be able to see the address signal on Z1 or Z2's output unless one of the keys associated with that output is pressed. With no key pressed, there is no voltage applied to the KR (Keyboard Row) lines. When a key is pressed, the associated pull-up resistor supplies voltage. Then you will be able to see activity on a KR line.

Input and Output Port

As you know by now, the TRS-80 microcomputer system is Memory Mapped. But it does have input/output ports. The basic difference in memory mapping and ports is in the method data is handled. In memory mapping, the CPU knows where the data is. In a port, the CPU does not know and couldn't care where the data is located. If the port is some kind of memory, the CPU will output data to that port; and it would be up to port circuitry to process and store data. In the input condition, the CPU accesses the input port; and it is up to the port to find data and feed this data to the data bus for the CPU.

The Z-8Ø CPU can access up to 256 output/input ports. In the TRS-80 Level I system, we only use one. The Cassette Recorder is the only port used. Its address is FF in hex. (Ports are accessed using only the lower eight address lines.)

Port Addressing

Since the TRS-80 uses only one output/input port, there must be only one port decoder. The port decoder is shown on Schematic Sheet 2, between the Sync Mixing circuit and the Power Supply.

Z54 monitors address bits Z1 through Z7. Z52, pin 5 monitors the AØ line. When hex FF is outputted on address lines AØ through A7, Z54 pin 8, and Z52 pin 6, will go low. These two outputs are tied to OR gate Z36, pins 2 and 1. When AØ* and FE* are low, FF* at pin 3 of Z36 will go low. The port address decoding is now complete. If we have a low at OUT* (the CPU wants to access an output port when this signal is low), Z25, pin 8 will output OUTSIG* because OUT* and FF* are low. If we have a low at IN* (the CPU wants to access an input

port), Z25, pin 6 will output a low generating INSIG* because IN* and FF* are low. IN* and OUT* should never be active (low) at the same time. Thus, INSIG* and OUTSIG* should never be active (low) at the same time.

OUTSIG*

The OUTSIG* line is used to control two cassette functions and one video function. It is used to generate the audio signal for the Cassette Recorder in a CSAVE condition. It is used to control the Recorder's motor also. Its video function is to control signal MODESEL (Mode Select). MODESEL will change 64 character format to 32 character format or vice versa. OUTSIG* is also controlling a latch made up of NAND gates in Z24. We will discuss this circuit later.

Z59 is a data latch controlled by OUTSIG*. This latch accepts data on lines DØ through D3. DØ and D1 are tied to pins 4 and 5 of Z59. These two inputs are used to output data that is recorded on tape during a CSAVE function. D2 is connected to pin 12 of Z59. This input controls the status of the Recorder's motor. D3 is connected to pin 18 of Z59. This input controls the status of MODESEL*.

The input to latch Z59 is stored and transferred to the output each time OUTSIG* goes high (rising edge triggered). For example, if input D2 is high when OUTSIG* goes high, pin 1Ø of Z59 will go high and stay high. The Recorder's motor will turn on. If input D2 is low when OUTSIG* goes high, pin 1Ø of Z59 will go low and the Recorder's motor will be off.

Cassette Motor Control

At the start of a CSAVE function, the Cassette Recorder motor must be turned on. The CPU will cause OUTSIG* to go low and apply a logical "1" to D2. When OUTSIG* goes high, the high on D2 will be transferred and held at pin 1Ø of Z59. This output is connected to relay drive Z41, pins 1 and 2. Pin 3 of Z41 will go low causing current to flow through relay K1's coil. K1's contacts close, shorting out pins 1 and 3 of J3. These two pins are associated with the remote jack at the Recorder. The Recorder's motor will then turn on.

Notice diode CR3 and Zeners CR9 and CR1Ø. CR3 is a standard silicon diode used for an "anti-chatter" function. When power is applied or removed from K1's coil, a counter EMF is generated. This voltage could be high enough to damage Z41's output transistor or cause K1 to click off and on a couple of times resulting in undue wear of the switching contacts. CR3 will shunt the counter EMF voltage around K1 and prevent transistor damage or relay chatter. Zener diodes CR9 and CR1Ø are used in somewhat the same way. Here we're trying to protect K1's switch contacts. When the Recorder is turned on, a high voltage spike could be produced, resulting in contact arcing. CR9 and CR1Ø prevent possible damage by shunting any voltage spikes above a certain level away from the contacts.

Cassette Audio Output

After the motor is turned on, the CPU may output data for storage on tape. All data timing for this output function is software controlled. Z59 is used to store data from the CPU and it "builds" the output waveform using CPU data. CPU data, under software control, is applied to latch Z59 on pins 4 and 5. Output pins 2 and 6 are connected to a resistor network consisting of R53 through R56. As OUTSIG* is clocking data into Z59, the resulting output on the line labeled CASSOUT resembles a sine wave built out of square waves. Figure 10 is an illustration of one bit time.

In Figure 10, the voltage output is a function of the status of pins 2 and 6 of Z59. In the period labeled T1, the output is shown as 0.46 volt. T1 is when output pin 2 is zero and output pin 6 is high (D0=0; D1=0). The voltage during T2 is outputted when pin 2 is high and pin 6 is high (D0=1; D1=0). The voltage during T3 is outputted when pin 2 is low and pin 6 is low (D0=0; D1=1). All "digital sine waves" are produced in this way.

Notice the time periods shown in Figure 10. From the start of one bit time to the start of the next bit time is two milliseconds. A one or zero is dependent upon the presence or absence of a pulse between the start of two bit-times. For example, when the CPU outputs a one-bit, it will generate a start pulse. One millisecond later, another pulse will be generated. One millisecond later, the start pulse of a new bit is generated. If this bit is to be a zero, then there will be a two millisecond delay before another pulse is generated; and this pulse starts the third bit-time. The pulses are outputted to the Cassette Recorder from pin 5 of J3. This pin is tied to the AUX input of the Recorder. The CPU outputs all of the instructions in system RAM to tape during the CSAVE function. When the function is complete, audio to the Recorder is disabled and a low is outputted at D2, shutting off the Recorder's motor.

Data is written on the tape in the following format: Upon CSAVE, the CPU forces Z59 to output 128 zero bits. It then outputs hex code A5 used by the CPU during CLOAD for synchronization. A two-byte starting address and a two-byte ending address is added next. Then the data follows, however long it is. After the data, the last portion to be stored on tape is the check sum. This one byte number is the sum of all data added together. It is used by the CPU to ensure what it CLOADed-in is what was CSAVEd-out. If the check sums don't match up, then there was a load error.

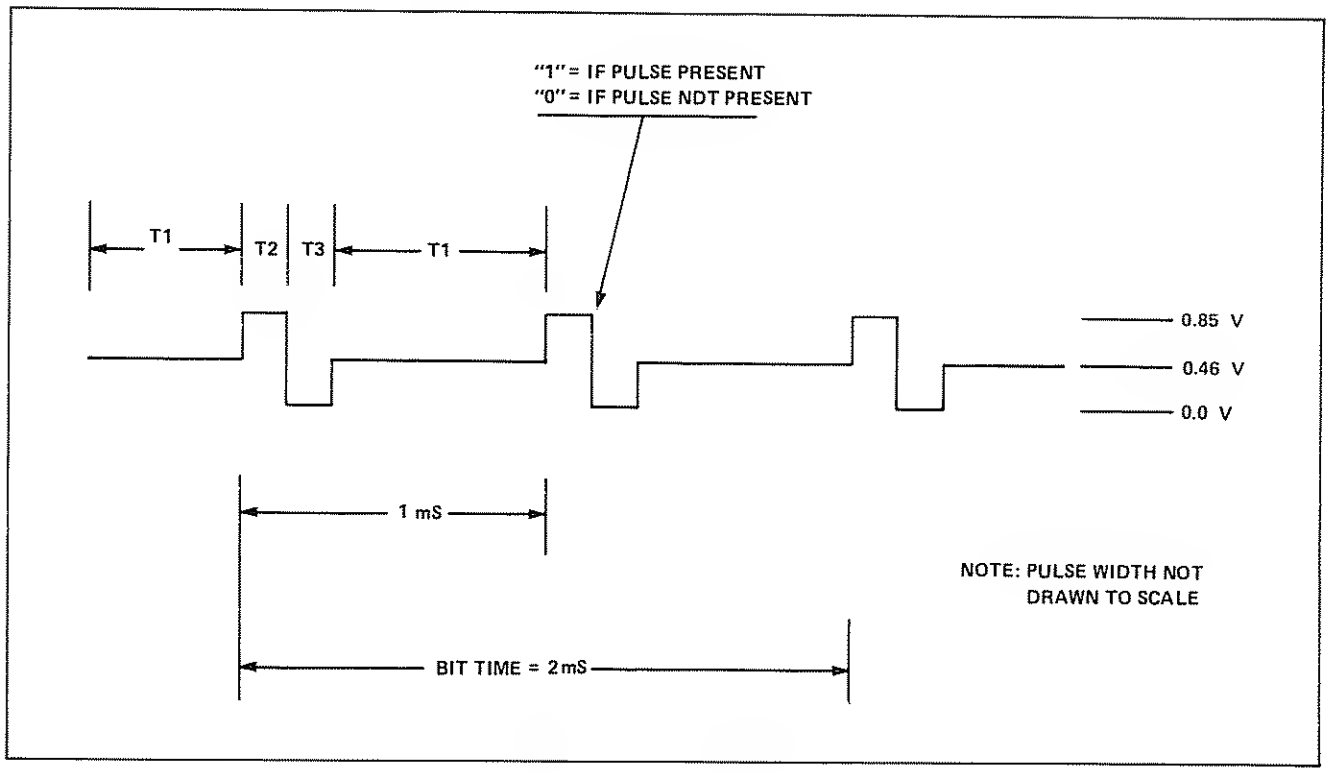


FIGURE 10. Idealized CASSOUT

Cassette Audio Input

If the Recorder could faithfully give back what was sent to it, we could eliminate a quad operational amplifier and a handful of passive components. But, it doesn't; so Z4 stays in. Matter of fact, the Recorder adds stuff to the signal. Motor noise and 60 cycle hum complicate signal processing considerably.

Upon a CLOAD instruction from the CPU, the Recorder motor turns on and cassette audio is applied to pin 4 of J3. This signal, called CASSIN, is tied to capacitor C24 and resistor R67 at the input of the audio processor section.Z4, pins 1, 6 and the output pin 5, form an active filter. This part of the circuit is used to filter out undesired noise and hum from CASSIN. It is a high pass filter, with about a 2 kHz roll off.

If you looked at CASSIN using an oscilloscope, you would see the data pulses riding atop a 60 Hz hum signal. After passing through the high pass filter, the resulting waveform would have the 60 Hz hum removed and only the data pulses would be left. The signals are swinging above and below a base line of about 2.0V. Figure 11 shows some idealized cassette signals. The signal drawn at Line A is the type that could be expected at the output of the active filter.

Once filtered, the next section of Z4 is used as an active rectifier. CR4 and CR5, together with the biasing resistors around pins 2, 3 and 4 will full-wave rectify the data pulses. A typical output on the cathode side of CR4 is shown on Line B of Figure 11.

After rectification, the signal is inverted and amplified. Z4, pins 8, 13 and 9, is wired to form an inverting amplifier circuit. The ratio of R41 and R42 gives the amplifier a gain of about 2. Line C in Figure 11 shows a typical output at Z4, pin 9.

The last stage of Z4 is used as a level detector. CR6 and CR7, together with C39, form a power supply of sorts. The amplified audio signal from Z4, pin 9, is applied to the anode of CR6. CR6 and CR7 decrease the voltage level of the incoming signal by about 0.8 to 1.0 volt. C39 filters the resulting voltage and creates a DC signal like the one shown on Line D of Figure 11. If the signal output from Z4, pin 9, drops below the reference voltage level at C39, Z4, pin 10 will go low. It will stay low as long as the voltage on pin 12 of Z4 stays below the reference. Line E shows the resulting output from Z4, pin 10. Notice that we lost a couple of pulses of audio because the signal did not swing toward ground enough to trigger Z4, pin 10. The negative transaction at pin 10 will be used to set flip-flop Z24. Cassette data will be converted into program data by the software in ROM and the CPU.

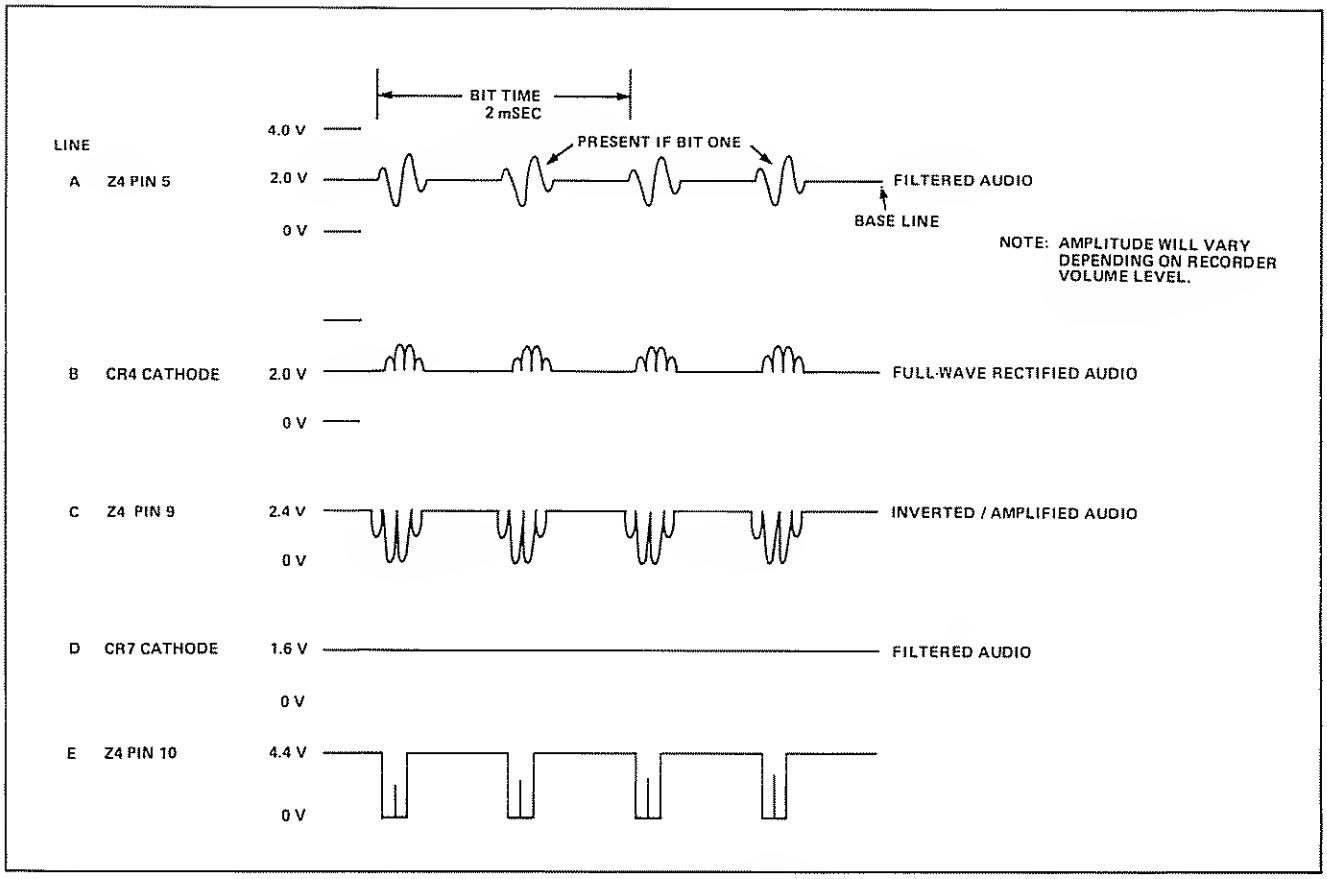


FIGURE 11. Audio Processing (Idealized)

INSIG*

Exactly how the CPU turns a string of ones and zeroes into the text of a 8ASIC program would interest only the hardcore software person. The amount of hardware used in the TRS-80 to get cassette data to the CPU is minimal. Only the hardware will be discussed.

Z25, pin 4, is tied to IN*. This signal will go low when the CPU wants to input data from a port. Port Addressing has already been discussed. A low at pin 4 of Z25 and a low at address decoder Z36, pin 3, will cause a low at Z25, pin 6, INSIG*. This signal is controlling only one device — part of Z44. Z44, pin 12, is tied to pin 8 of NAND gate Z24. The two NAND gates of Z24 are wired to form a set-reset latch. If pin 9 goes low, pin 8 will go high. Pin 8 is cross tied to pin 12. If pin 13 is high and since 12 is high, pin 11 will be low. With a high at pin 8 and a low at pin 11, the flip-flop is "set". If pin 8 is low and pin 11 is high, the flip-flop is "reset". The flip-flop is being set by cassette data and reset by OUTSIG*. Z44 monitors the status of Z24 under command from INSIG*.

Here is how it works during a CLOAD function: When CLOAD is entered via the Keyboard, OUTSIG* will go low, starting the Recorder's motor and resetting Z24 by pulsing pin 13 low. The first time Z24, pin 9 goes low, the first bit time starts. This is shown in Figure 12 at Line A. Line D, the output of the latch, goes high as soon as pin 9 goes low. OUTSIG* goes low after a short time delay, shown on Line C. The signal will reset the flip-flop as Line D shows. A short time after OUTSIG* goes back high, the CPU will test Z24, pin 8's status by enabling Z44. Line D is low at this time. The CPU recognizes a logical "Ø" during bit time 1 as shown by the Ø under Line D. The next time Line A goes low is the start of bit time 2. The low on Z24, pin 9, sets the flip-flop. OUTSIG* resets the flip-flop a short time later. INSIG* then enables Z44 and checks the status of the flip-flop. The CPU "sees" a zero again, so bit time 2 is a zero bit. The next low on line A starts bit time 3. It sets the flip-flop, and a short time later OUTSIG* resets the flip-flop. Before INSIG* can test status, another low comes from the audio processing level detector and sets the flip-flop. Now INSIG* goes low, checking status. It finds Z24, pin 8, high. The CPU labels bit time 3 a "1". Now the CPU must reset the flip-flop before bit time 4 starts. Line C shows the added OUTSIG* pulse to reset Z24. The flip-flop is reset and stays reset until the next low on Line A sets it again. The CPU finds bit time 4 to contain a zero. This set/reset process continues until the CPU has read every bit-time of the program that was stored in the cassette. It is the CPU's responsibility to assemble the bit times into data words, the words into text, and store the text in RAM. The CPU is quite busy during a CLOAD function.

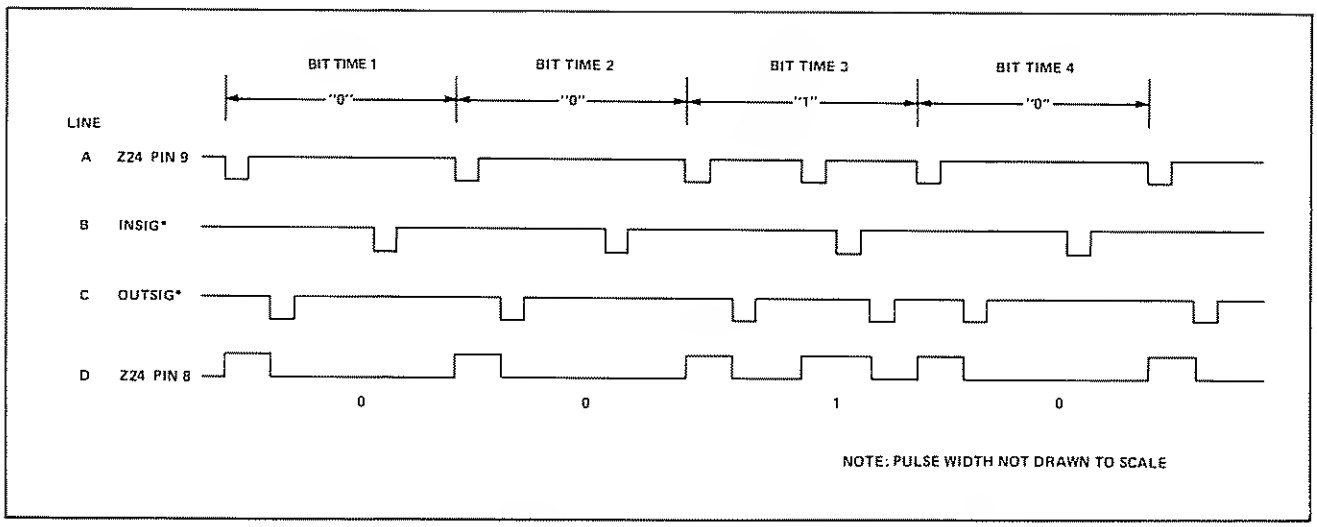


FIGURE 12. Data Latch Timing

System Power Supply

The TRS-80 needs three voltage levels: +12 volts at about 350 milliamps; +5 volts at about 1.2 amps; and —5 volts at 1 milliamp. The +12 and —5 volts are needed by system RAM and everything needs +5 volts. The +12 volt and +5 volt supplies are regulated and current-protected against shorts. The —5 volts supply is not as critical as the other two supplies, and it uses a single zener diode for regulation. Stepped-down AC voltage is supplied to all regulator circuits from a UL approved "AC adapter."

AC Adapter

The AC Adapter is a large version of the type used for calculators and TV game products. Inside the plastic case is a single transformer with one primary and two secondary windings. The primary circuit is designed for 120 VAC and has an operating range of 105 to 135 VAC.

NOTE: There is a wire fuse in the primary side to meet UL specifications.

The secondary windings are both center-tapped. One secondary is rated at 14 volts AC at 1 amp. This winding is used for the +5 and —5 volt supplies. The other secondary winding has diodes connected and it outputs 19.8 VDC at about 350 milliamps. This circuit is used for the 12 volt supply. All voltage outputs and center taps are brought into the POWER input (J1).

+12 V Power Supply

Unregulated DC voltage for the +12V supply is inputted at pin 2 of J1. When power switch S1 is closed, C8 filters the voltage and the net result is approximately 20 volts, which is applied to Q6 and regulator Z2. Figure 13 shows a simplified diagram of the internal circuitry in a 723 regulator chip. Figure 13 will help in the regulator operation discussion.

The filtered DC voltage from the Adapter and C8 is applied to pin 12 of Z2 and the emitter of series pass transistor Q6. The voltage applied to pin 12 allows a constant current source to supply zener current for Za. Pin 6 of Z2 will output a zener voltage of about 7.15 volts. Pin 6 is tied to pin 5, the positive input to operational amplifier Zb. The negative input to the op-amp is tied to the wiper of R10. Initially, pin 4 of Z2 is at ground, forcing the output of op-amp Zb to output about 7.15 volts. Transistor Qa turns on, which turns on pass transistor Q6. The pass transistor supplies voltage for current monitoring resistor R18 and to the resistor network R13, R10 and R12. If R10 is adjusted for 7.15 volts at its wiper, the op-amp will be balanced and Q6 will output only enough voltage to keep the loop stable. If output voltage dropped below 12 volts, Zb's output would decrease which would force the current through Qa to decrease. Qa would cause Q6 to increase the current through it, and the output would rise back up to the 12 volt level. If the 12 volt line increased in voltage, the op-amp would cause Qa's current to increase, forcing Q6 to drop down.

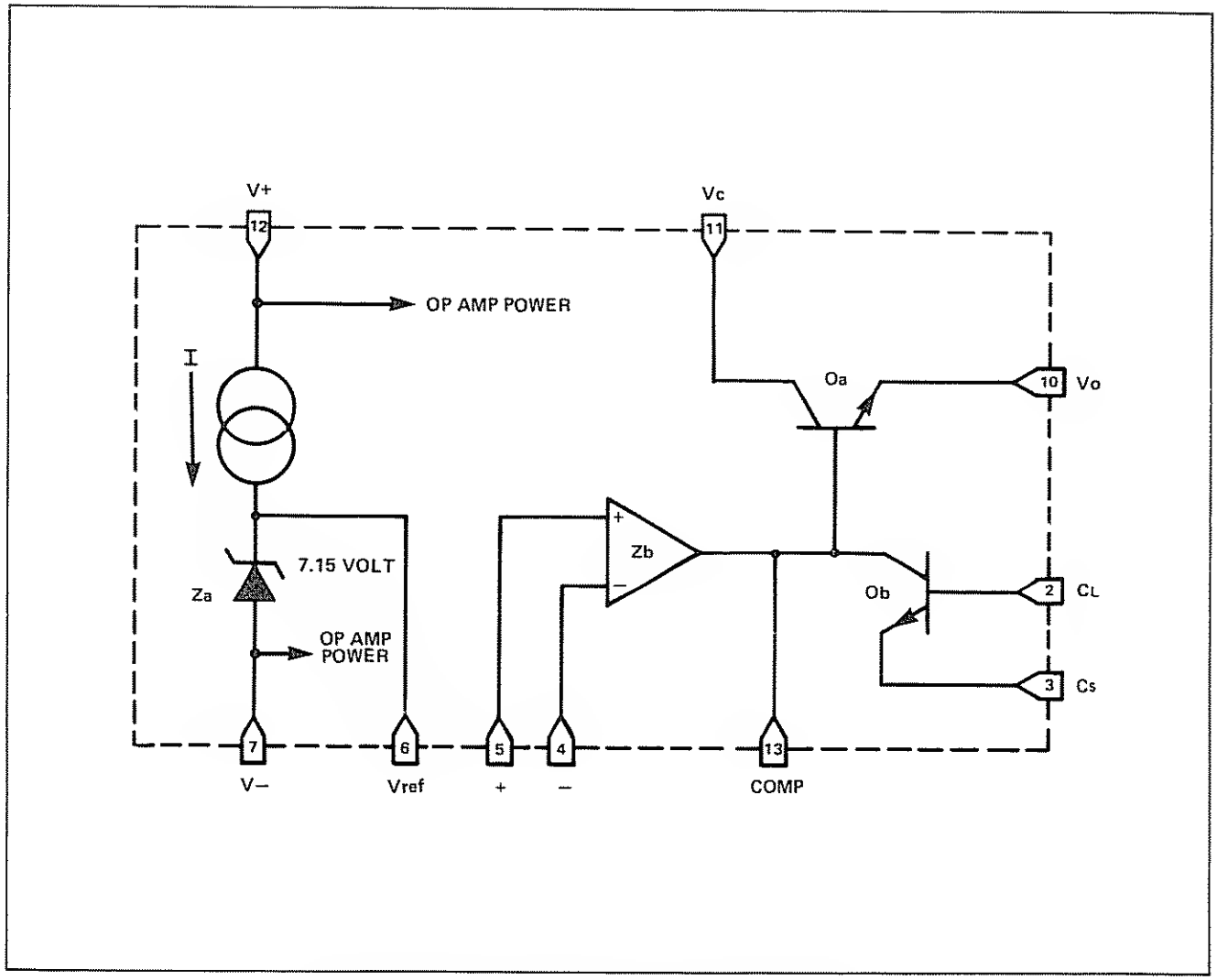


FIGURE 13. Block Diagram of 723 Regulator Z2

The transistor labeled Qb in Figure 13, is used to protect power transistor Q6 against over-current damage. If R18 drops sufficient voltage to cause the resistor node at Z2, pin 2, to reach 12.6 volts, Qb will take command of Qa. As Qb is turned on, Qa turns off which starts turning Q6 off. The voltage at Z2, pin 10, must approach 14.7 volts before Qb takes charge of Qa. 14.7 volts at pin 10 means that the 12 volt supply is approaching its maximum design current of 480 milliamps. If a short develops across the 12V supply, Qb will activate, forcing Qa to shut down. With Qa off, Q6's base rises to the input voltage level because of R16. Q6 snaps off the supply, preventing it from attempting thermal suicide. Once the short is removed, Qb will turn off and the system will operate normally.

Capacitor C13, connected between pins 13 and 4, is a frequency compensation capacitor. It prevents the op-amp loop from going into oscillation. C11 and C15 are the supply's output filter and noise suppressors. Capacitors C28, C30, C32 and C34 are distributed along the 12 volt supply bus for transient suppression.

+5 Volt Supply

The 5 volt power supply also uses a 723 regulator. Due to the current and voltage requirements, more components were stuck around the regulator for support. But the basic circuit operates the same. Figure 13 will also be used in this circuit.

For the 5 volt supply, the AC adapter supplies about 17 volts AC at J1, pins 1 and 3. Full-wave rectifier CR8, rectifies the AC. When S1 is closed, about 7 VDC is passed through the switch contacts and is filtered by C9.

The power supply for Z1 and the current source for zener Za is taken from the regulated side of R18 in the 12 volt section. Pin 7 is grounded as in Z2, but the zener output is handled differently. The 7.15 volt zener voltage is applied to the resistor network consisting of R6, R5 and R11. When R5 has been adjusted for a 5 volt output on the supply bus, pin 5 of Z1 will be at about 5 volts. The negative input of the op-amp, Zb, is sourced through a 1.2K resistor, R7, and tied to the 5 volt bus. The op-amp controls Qa, which controls bias drive for Q3. Q3 is used to handle the greater base drive necessary for pass transistor Q4. Q4's collector is tied to current sensing resistor, R4. R4 monitors the current the 5 volt bus is producing just as R18 did for the 12 volt bus.

Circuit operation is exactly the same for Z1 as it was for Z2. If op-amp Zb detects a rising or falling voltage condition at the output bus, it will adjust base current to Qa. Since Qa cannot handle the drive requirements for Q4 directly, Q3 is needed for current gain. During current limiting condition, Q5 monitors the voltage across R4, which is a direct function of bus current. As Q5 begins to turn on, the node at R3 and R9 supplies more voltage for base drive of Qb. As Qb takes command of the regulator loop, Qa is commanded to start cutting Q3 off. Q3 begins to turn Q4 off and the circuit goes into current limiting. The current limiting action of Q5 starts to come into play when the voltage across R4 approaches Ø.6 volt. Ohm's Law tells us the bus current at this voltage level is approaching 1.82 amps.

C12, connected between pins 13 and 14 of Z1, performs the same compensation function as C13. CIØ and C14 are the output filter and noise suppressors while the thirty-two Ø.ØI microfarad capacitors are distributed all over the Board to suppress transient spikes.

Notice zener diode CR1 on the 5 volt bus. This diode is used as crowbar circuit protection in case of catastrophic failure in the RAMs. If something happens in the system RAM circuit that causes a short between the 12 and 5 volt buses, CR1 would turn on, causing the 5 volt bus to go into current limiting. Since CR1 is a 6.2 volt zener, it would protect the TTL devices connected to the 5 volt bus from being damaged by a sudden 12 volt supply voltage. Normally, CR1 would be off with no current flowing through it.

NOTE: The 12 volt supply must be working properly before the 5 volt supply will operate correctly. Therefore, the 12 volt supply must be adjusted before the 5 volt supply.

−5 Volt Supply

Source voltage for the −5 volt supply comes from the negative terminal of rectifier CR8. When switch S1 is closed, the negative DC is filtered by C1 and about −11 volts is applied to resistor R19. R19 is used to limit current for zener regulator CR2, a 5.1 volt device. The −5 volt circuit is about as simple a power supply as can be designed. C4 and C3 is the −5 volt supply filtering and noise suppressing caps, while C16 through C19 perform the transient suppression function.

Level II ROMs

One of the most fascinating aspects of a computer is its versatility. It can do almost anything; all it needs to know is how. With a single ROM change, the TRS-80 can speak in any higher level language we might care to use. In Level I we had just enough mathematical and symbolic capabilities to inspire bigger and better things. The difference between a Level I TRS-80 and a Level II TRS-80 is inspiration and a bigger ROM. Level I uses 4K of ROM. In Level II, the system uses 12K of ROM. The added 8K of ROM is enough to give text editing; transcendental functions; give it giant numeric and string arrays; and more system variables than you can use. The guts of the system have not changed; the hardware is the same. The only difference is the machine language contained in ROM.

A Level II machine can be identified by a separate ROM Board stuck to the etch side of the CPU board. This Board contains three 4K ROMs, a TTL decoder, and a ribbon cable. The cable attaches the majority of the ROM's address inputs and all of the data outputs to the now empty main Board ROM sockets. There is also a 4-conductor ribbon cable (green, orange, red and yellow) coming from the Level II Board. The conductors connect to the CPU Board at A11, A12, A13 and ROM*. The conductors enable a Level I to Level II conversion on any level production Board to be quite painless.

A11 is used as an address for all three of the ROM's. It is tied to pin 18 of Z1, Z2 and Z3. A12 and A13's leads go to the AØ and A1 inputs to decoder, Z4. Z4 is an addition to the address decoder network on the main Board. When A12 and A13 is ØØ, pin 1 of Z4 goes low and ROM A is enabled. When A12 and A13 are Ø1, ROM B is selected; and when A12 and A13 are 1Ø, ROM C is selected by a low at pin 3. Since we want the ROM's to be accessed only when the CPU needs instruction, ROM* is therefore brought into Z4 to act as a master enable. Only when ROM* is low will we ever select ROM A through ROM C.

ROM power, minor ROM addressing and data output are handled by the large ribbon cable. One end of this cable is connected to the socket of the ROM Board while the other end is attached to Z31's socket. Addressing and data output are handled by the same circuits that support Level I.

Adjustments & Troubleshooting

Disassembly

1. Position the Computer, keyboard down, on a soft surface.

2. Remove six screws from the bottom of the Case. Keep track of the different lengths so you are sure to replace them in the correct holes.

3. Turn the Computer right side up and carefully separate the case halves.

NOTE: There are two different types of LED mounting

A. Keyboard PCBs which are double-sided, with plated holes use an LED mounted into the top half of the Case. Long wires connect the Keyboard PCB and the LED. Slip the ring off the LED socket body (using needle-nose pliers). Allow the ring to slide down to the Keyboard PCB. Use the eraser end of a pencil to push the LED down through the plastic Case. Bend the LED leads slightly to prevent the retainer ring from getting lost.

B. The second mounting technique has the LED soldered directly to a single-sided Keyboard PCB. The top half of the Case simply fits over the Keyboard and the LED.

4. Set the Case top aside. Carefully lift the Keyboard out of the Case bottom. Take care that you don't strain the Interconnect Cable.

5. Notice the five rubber spacers which separate the two PCB's. Keep in mind where they are positioned so you replace them over the same plastic studs.

6. Carefully lift both boards out of the Case. Take care you don't strain the interconnect cable.

7. Carefully lay out the two PCB's, side-by-side on your working surface. Again, avoid putting strain on the Interconnect Cable.

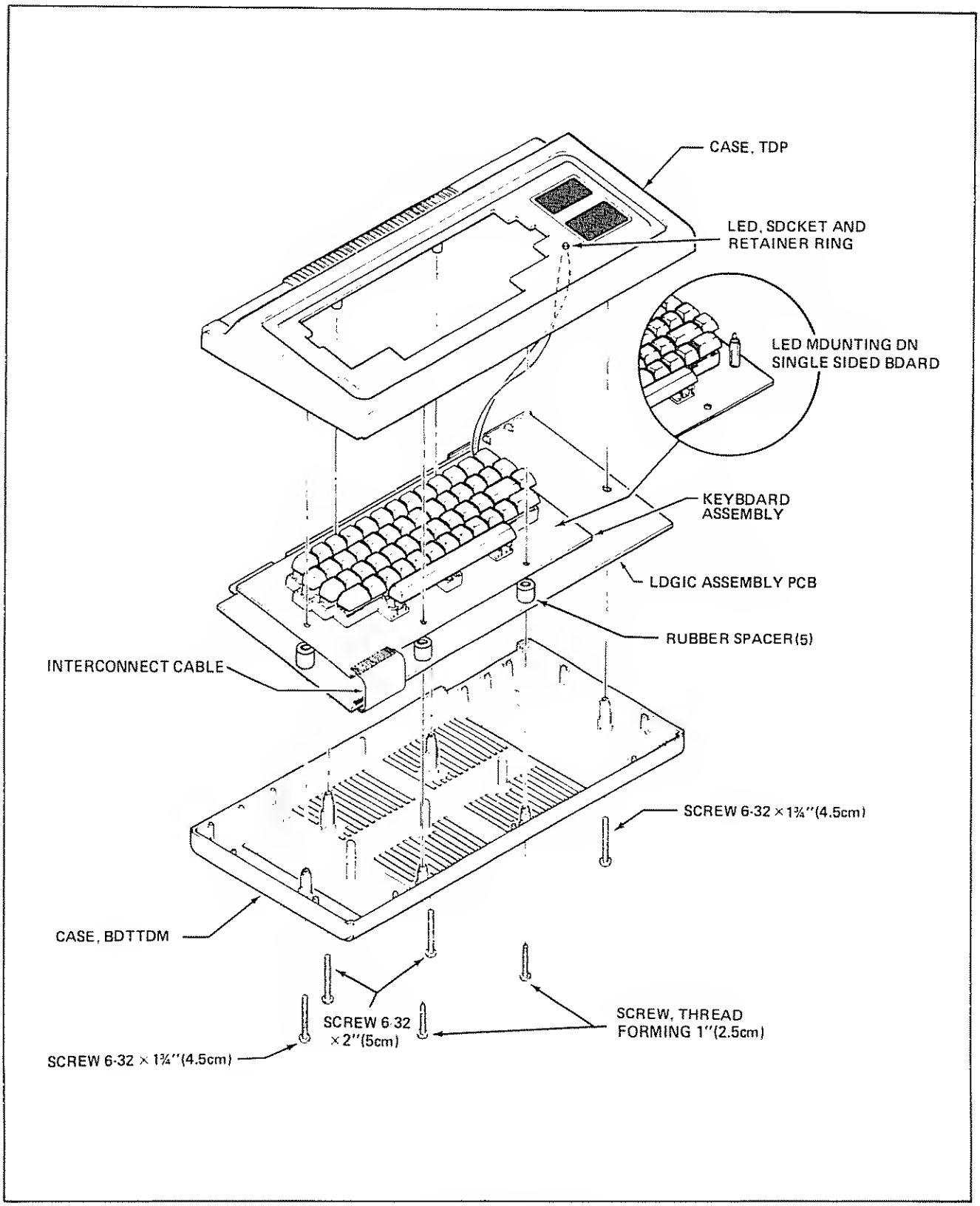


FIGURE 14. Exploded View

Power Supply Checks And Adjustments

Once the PCB's have been removed from the Case and are resting on the test bench, connect the Power and Video DIN plugs.

CAUTION

The CPU Board is now "upside down" in reference to its normal position in the case. Be sure you connect the Power DIN plug to the Power jack (J1) and not to the Cassette jack (J3). The Power jack is the one closest to the Power switch.

Turn on power to the CPU Board and the Display. Data may or may not be present on the Display depending on the type of problem. Disregard the Display for the time being. Test power supply voltages first (see Figure 15).

1. Connect the common (—) lead of a Digital voltmeter, to the right side of capacitor, C9 (that's the largest capacitor on the Board).

2. 12 VOLT SUPPLY. Measure the voltage present at the top side of power resistor R18. (The "top side" would be the end closest to the edge of the PCB.) Voltage should be 12.Ø volts ± 5% (12.6 to 11.4 volts). If the voltage does not fall within these limits, adjust R1Ø for a correct reading.

3. +5 VOLT SUPPLY. Measure the voltage at the left side of R4. (This 1 watt resistor is located between the two large electrolytic capacitors, C8 and C9.) Voltage should be 5.Ø volts ± 5% (5.25 to 4.75 volts). If the voltage does not fall within these limits, adjust R5 for a correct reading.

NOTE

Do not adjust the 5-volt supply until the 12 volt supply has been checked and is within tolerance.

4. Measure the voltage at the anode of CR2 (CR2 is located to the left of the Power switch). Voltage should be —5 volts ± 5%. There is no adjustment for the —5 volt power supply. If this supply fails to fall within the voltage range, you must isolate the problem to a defective component(s).

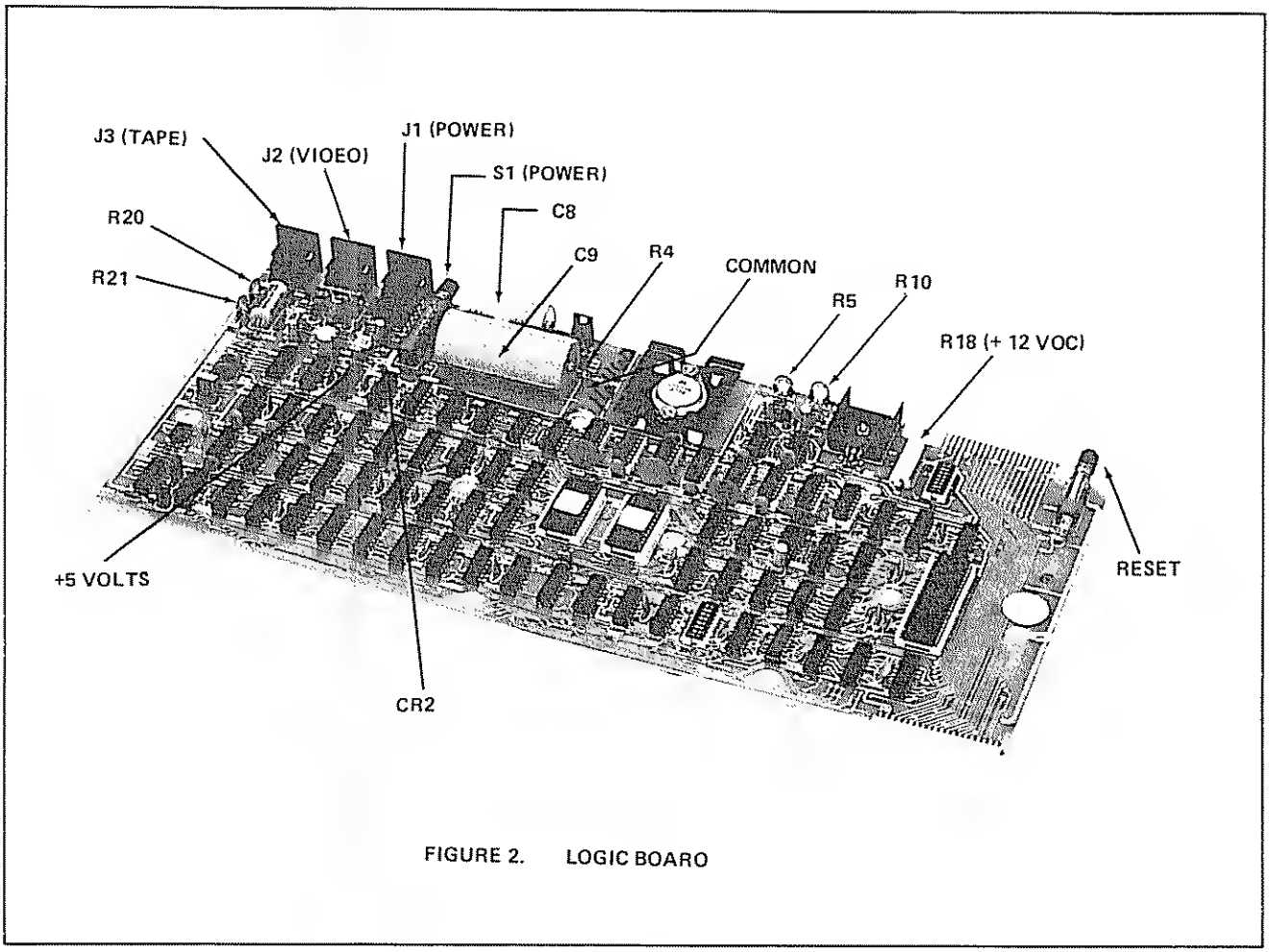


FIGURE 2. LOGIC BOARO

FIGURE 15. Logic Board

Section Isolation

Usually, problems or complaints are directed to a certain function. For example: CLOAD is fine, but when LISTing the program, half of the listing is a screen full of junk. Since part of the LISTing is correct, we can assume the audio processing circuitry works. You might suspect:

1. A problem with tape data or.
2. a RAM error is screwing up the data input.

You might listen to the tape's audio for voids. or attempt to load a test program and exercise the RAM's. In either case, the problems will give you some clues where to look. The next step is to eliminate the suspect areas and narrow it all down to a bad component or connection.

One of the hardest problems to Section-Isolate is: A screen full of junk on power-on. (A display with all character positions filled with either alphanumerics or graphics.)

As you read in the theory of operation section, at power-on the CPU exercises an initiation procedure. A "garbage condition" results from the CPU bypassing this routine as it goes crashing through any other routine it finds. Unfortunately, a garbage condition does not always indicate that the power-up logic is defective. A problem could exist in RAM, ROM, the video divider chain and/or, of course, the CPU itself. Therefore, a problem could exist in 75% of the Computer.

Where do you start? You could start replacing everything that's easy to get out: RAM, ROM and the CPU. But, you are really wasting time. If the problem is a simple solder short, replacing all socketed devices is not going to help. There is a method you can use for Section Isolation. It is based on a removal technique that eliminates sections from the suspect list.

Section Isolation Flowchart

Figure 16 is a Flow Chart of Section Isolation "by part removal." You start the process in the parallelogram, block 1. This is the basic problem. Block 2 instructs you to disassemble the unit and reconnect the video and power inputs. Block 3 is a decision block. Do you have garbage on the screen now? If so, you continue to block 5. If not, block 4 tells you to suspect a shorting Interconnect Cable between the Keyboard and the CPU Board. You could also have loosened a "solder ball" during disassembly, and a short is now gone. Examine the Interconnect Cable carefully for shorting conductors. Did anything fall out of the Computer during disassembly? You might have fixed the Computer just by taking it apart! Test it again.

At block 5, you will turn off power to the unit, wait about ten seconds, then reapply power. The delay gives the initialization logic time to reset. If there is now a "READY" on the screen at block 6, maybe you have a problem around S2 or C42, as block 7 instructs.

In block 8, you are instructed to remove the DIP shunt (X71) at Z71. With X71 removed, the RAM's are not electronically in the system. When power is applied, the ROM and the CPU are in communication, but there is no data flow to or from RAM. The screen should show a pattern of 16 character lines of 32 colons. If the CPU shows large colons, you could have RAM problems or keyboard-type problems. Blocks 11 through 15 will help in isolating that type of problem. As blocks 12 and 14 imply, there are two colon displays. One display is stable. The other is blinking and flickering as the CPU constantly interrupts video addressing. Depending on the status of the Keyboard, you could have data line or keyboard problems.

The next step at block 16 is to remove the ROM's. The CPU is now locked up without instruction from ROM. The pattern to look for is

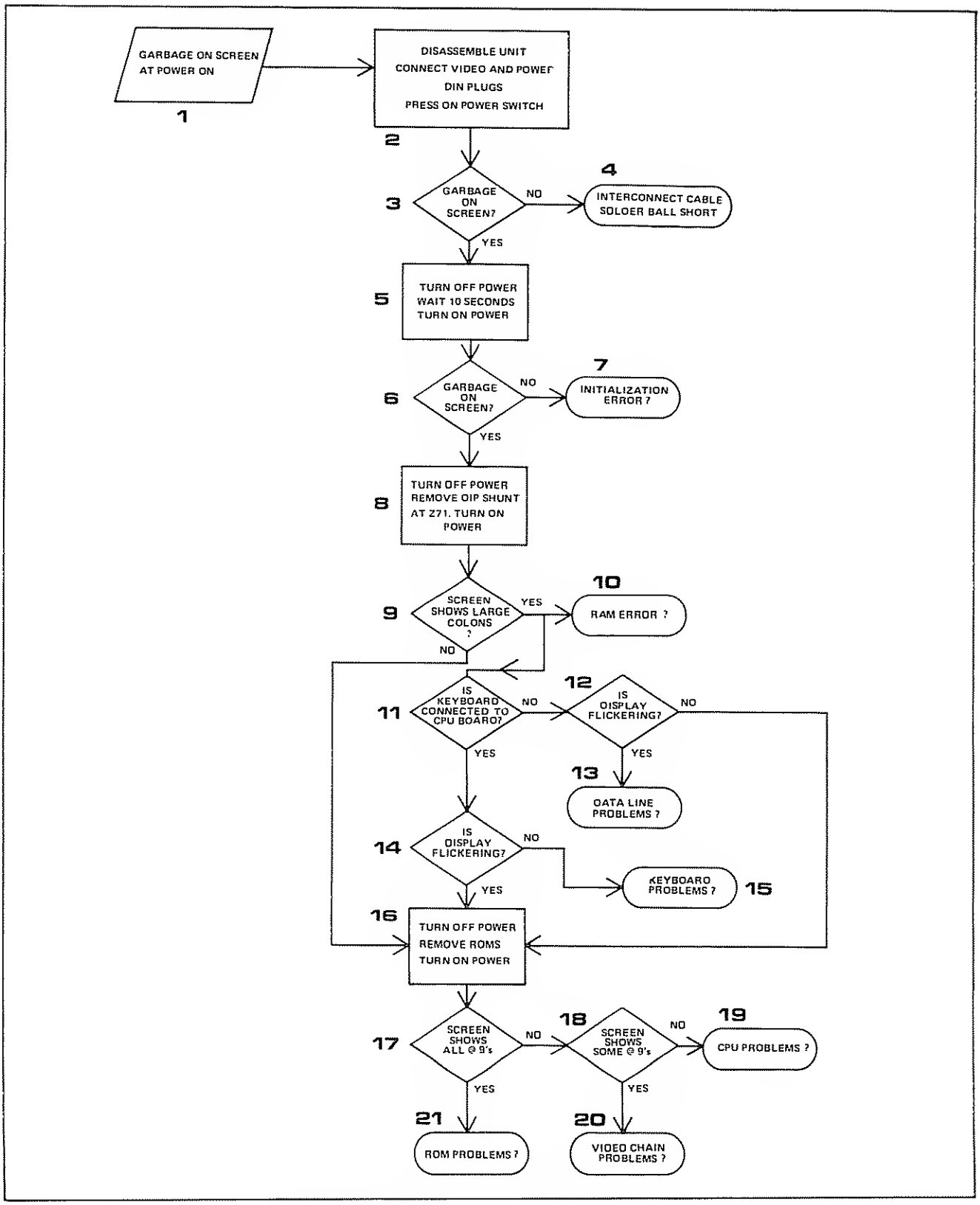


FIGURE 16. Section Isolation Flowchart

a screen full of @9's. The Display should be in 64 character format at this time. The Display will flicker as VID* continually accesses the video ROM's.

If you get @9's on the screen, you probably have a ROM error. If no @9's or partial @9's are visible, you could have video chain or video RAM problems. If you still get garbage, maybe the CPU is dead or something is making the CPU not function.

As you can see, the part removal isolation technique uses a lot of maybe's, question marks and could be's. The "what if's" are trying to tell you, "I don't work all the time". You could have ROM problems and yet get large colons. You could get @9's and still have CPU error. But it's better than nothing, and the process does give a starting place.

Signal Condition

Activity, Steady State, Or Floating

Normal troubleshooting techniques call for an output-to-input sweep of the bad signal line. In other words, once a bad signal is found, the circuit is traced backwards until the signal is correct. The failed device will be located between the good input and the bad output. We will use this "backward" approach to isolate the defective components in the TRS-80. But, we will not attempt to analyze inputs and outputs unless we are between that magical good-in/bad-out point. What we'll do most of the time is check for activity or status of signal levels. If it has no activity or status, it must be floating.

Activity is defined as any logical transition from high to low or vice versa. For example, the output of oscillator buffer Z42, pin 6, always has activity. There is a constant output pulse train at this pin. The signal swings from almost ground to over 3 volts continuously.

Steady state is defined as a logical 1 or logical Ø. For example, Z4Ø, pin 16, has a steady state logical 1. It is held high by resistor R5Ø. Another example is the logical Ø at pins 6 and 7 of Z56, the CPU clock divider. Z42, pin 8, is always low unless resistor R67 is grounded.

Floating is defined as a signal level between the steady state of a logical Ø and a logical 1. The CPU, the ROM's, the RAM's and the data and address buffers are all tri-stable devices. When tri-state type parts are disabled or unselected, the output may show a floating condition. In a floating condition, the output will show system noise flickering through it. The average level of the noise will attain a voltage of 1.5 volts or so. TTL devices define a logical Ø to be equal to, or less than, Ø.8 volts. A logical 1 is defined as a voltage level equal to, or greater than, 2.4 volts. Any voltage between these two levels will be considered floating.

A floating signal may be "finger tested". If a finger is placed on a floating signal, the amplitude of the signal will increase radically. The noise floor is said to increase. An example of noise floor: If a ROM is disabled, the data output pins will be floating. The noise floor will show an average of 1.5 volts or more. If a data output pin is disconnected from the socket and then checked, it may have no noise floor and may look like a logical low. A quick finger test of the pin will cause the noise floor to increase rapidly. In both cases, the pin was floating. But, when the pins were disconnected from the socket, the ROM lost its noise source and looked low. The finger supplied a new noise source.

CPU

A problem with the CPU does not mean that Z4Ø is inoperative. It could mean that you have difficulty with the address and data buffers, the control group, CAS/RAS timing, or with one of the CPU's support devices. If you think you have a problem with one of these devices, you might substitute a **known** good CPU for Z4Ø for a quick check. But, chances are that the problem exists in another place. Don't assume that Z4Ø is

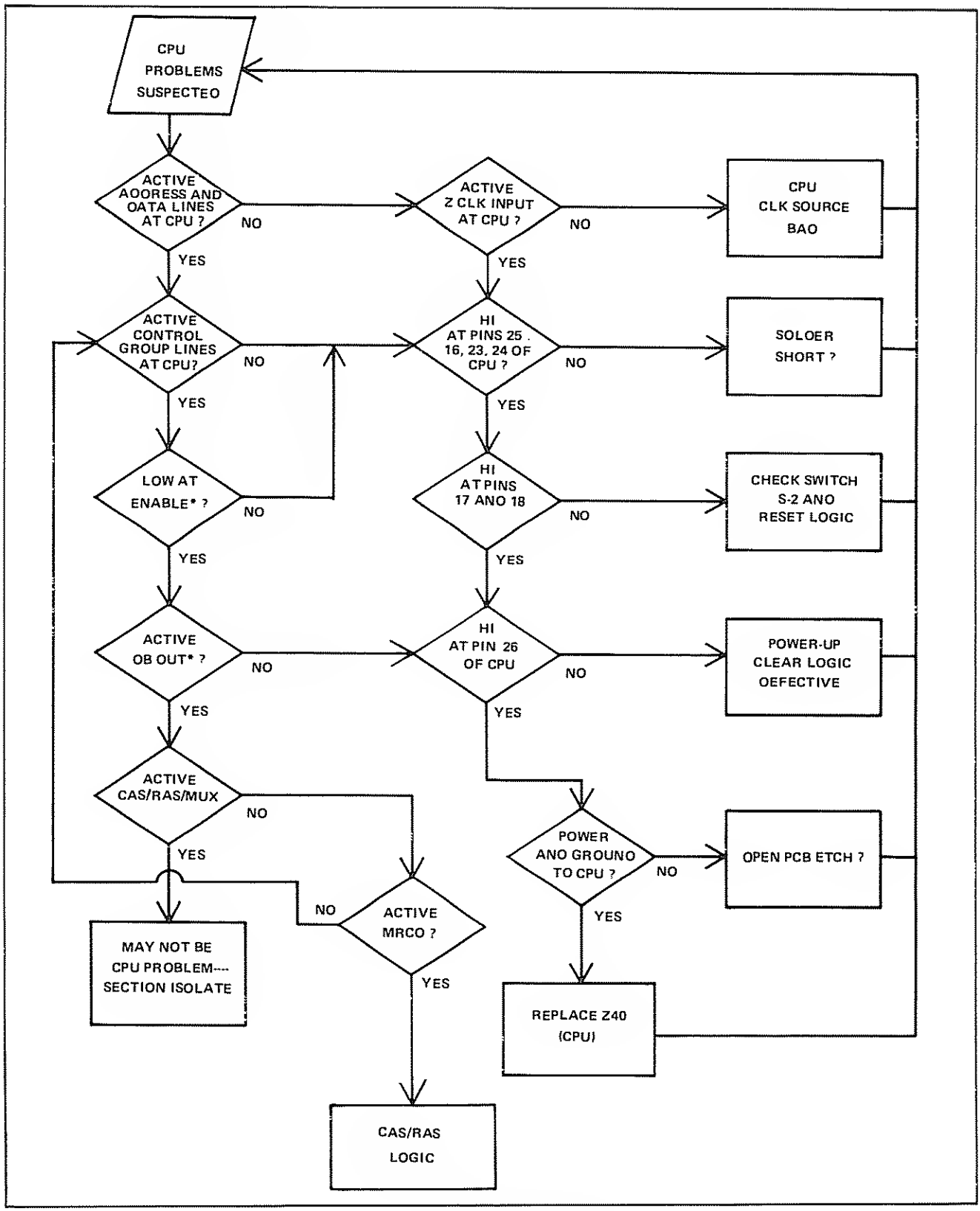


FIGURE 17. CPU Flowchart

trouble-prone just because it's in a socket and is easily replaced.

The flowchart, shown in Figure 17 will help in CPU troubleshooting.

The primary objective of this chart is to help you quickly find a signal that should be active but isn't. The main flow of the chart is on the left side of the Figure. Here, you are checking for activity on address and data lines. With no activity on the address lines, you are immediately branched off to the CPU's support group to find out why. Pay particular attention to the appearance of address line outputs. Any tri-state looking signal could mean a potential short between address lines. The opposite is true of data lines. These signals may be active and have floating components between active states. Hence, data line shorts are extremely difficult to find; using an oscilloscope.

If you need to disable the CPU for short checks, ground pin 25 of Z4Ø (or the side of R58, whichever is easier). The address lines will float on the CPU side, and the address buffer will be tri-stated. On the data lines, the output data buffers will be disabled, but the input buffers to the CPU will be enabled. Since the inputs to these buffers should be floating, the outputs will all appear high. You can check for a high at each output. If you want to see if the buffer operates, inject a TTL signal into the floating input and look on the output for that signal. You have a lot of signal sources in the video divider chain to choose from. You should see the activity of your injected signal without any floating components. If an output appears to have a floating condition, look for CPU-side shorts with other data bus outputs.

ROM

Problems associated with the ROM's can be broken down into three areas. You can have:

addressing
data
or chip select problems.

Addressing problems can be associated with open or shorted address lines going to the ROM sockets. Early versions of the Boards may have jumper modifications on the solder side of the Board that have broken loose. There is also the chance that vibration has jarred a ROM partially out of its socket. The address lines should be checked at the chip. Normally, there will be activity on all lines. You can also use the TEST* signal in the CPU section to disable the address lines and look for shorts on the address bus.

There are two types of **data problems.** The first is the non-repairable bit error internal to the ROM. The checksum contained in the SCQATS† program can readily verify this. If the ROM problem is too severe for SCQATS loading, a replacement test may be necessary. The second type of data problem is the short or open on the data output. If you remove DIP shunt Z3, the ROM's will tri-state and you can check for a floating state on the data pins.

Chip select problems are usually associated with ROM* or MEM*. ROM* is the chip select for one or both ROM's, while MEM* controls the data buffers for a ROM/RAM Read. You get both signals from DIP shunt Z71. You might check the DIP shunt for correct programming jumpers. (The type of DIP shunt used in the TRS-80 has been known to develop cracks in the shorting bars during programming. Metal cracks are very likely to be present if the plastic part of of the shunt is damaged.)

RAM

CAUTION

When handling RAM's, be careful that you do not damage the parts because of static discharge. Before touching a RAM, ground yourself using a grounding strap designed for handling MOS devices — or momentarily touch the right side of capacitor C9. If you must move MOS devices from one part of a room to another, be sure you have the parts

†SCOATS is a special machine language test and debugging tape available on special request from Radio Shack's National Parts Center.

in a conductive tube or in conductive foam. **DO NOT USE STYROFOAM!** Unless specially treated, styrofoam eats MOS devices like candy. It can generate tremendous static charges. **Do not use cellophane tape to hold RAM's in sets of eight.** The process of removing tape from the roll will act as a handheld static generator.

RAM problems are slightly more difficult to troubleshoot because of multiplexing of the address inputs. Aside from addressing differences, the RAM's are checked like the ROM's. Use SCOATS to dig out memory errors if possible. Also, be sure you check the three power supply voltages going to RAM; they are: −5, +5 and +12 volts. Insure that all voltages are present on all RAM's. Also check for bad RAM insertion — pins not in sockets or loose RAM's due to shipping vibration. Check for activity on RAS, CAS and MUX going to RAM and multiplexer. Insure that all specified addresses do indeed go to the multiplexer. RAM problems are most likely suspect after a 4K to 16K conversion. Be sure DIP shunt Z3 is programmed correctly for the amount of RAM in the system.

If you have a RAM problem and the system will not load SCOATS, you can replace the eight RAM's with a known good set. If this fixes the unit, start replacing your standard RAM's with the parts you took out, one by one. Power up after each exchange to see if you still have a "Ready". Continue this process until you have isolated the bad RAM(s).

Address Decoder

A problem in the address decoder section will probably point you in the memory direction. For example, if the ROM is never addressed with ROM*, you'd think you have ROM difficulties. Therefore, if you suspect one of the "memory" locations, keep in mind that the address decoder sources the memory selects. The select inputs to the different memories should be the very first thing you check.

Failure of address decoding will usually be associated with one of the higher order address lines. You should check Z21's inputs for activity, paying particular attention to pins 14 and 2 of Z21. Z21's outputs also depend on the status of the shorting straps on DIP shunt Z3. Remember: You cannot test an open collector output unless there is a pull-up resistor attached to it.

If Z21 and the DIP shunt appear OK, look for activity on the other inputs. For example, A12 and RD*. Maybe the CPU is constantly addressing one memory all the time, but is not getting any data in return. This is especially true of the keyboard "memory". If the CPU thinks there's a key pressed, it may lock itself into a loop, trying to isolate a phantom key.

Keyboard

Difficulty with the Keyboard is usually mechanical. Sticking keycaps, bouncy keys and a broken Interconnect Cable are common. Shorts in the Keyboard matrix are usually easily detected. If you find an alphanumeric character displayed right after the > , that particular key, or PCB run, may be shorted. A completely "dead" Keyboard could be caused by lack of power, a broken Interconnect Cable or the address decoder is not supplying KYBD*.

If you have a weak space bar spring, replace it with a 3 oz spring. If it still sticks, and there is no plastic flash that could cause sticking — give the spring more muscle by stretching it a little.

If you lift a sticking keycap and find mangled switch contracts, don't scrap the whole keyboard, replace the contacts. The following is a step-by-step procedure for contact replacement:

1. Disassemble the unit.

2. Remove the keycap and keycap plunger.

3. Remove the spring.

4. Unsolder the contacts from the PCB. Be sure that the contact ends (protruding through the Board) are actually free. This is important when you are working on double-sided Boards.
5. Note the position of both types of contacts. The fingered contact is usually on the right side, with the keyboard oriented in a normal position.
6. Using a pair of strong needle-nosed pliers, pull both contacts out of the plastic base. Fine pointed needle-nose pliers usually cannot grip the thin contacts well enough for extraction.
7. Insert a new contact set in tool #773-1ØØØØ or #773-1ØØ23, making sure that the contact fingers face each other.
8. Insert the tool with the new contacts into the key base. Press the tool firmly until it seats against the stops. Insure that the PCB is not resting on a hard surface because the tool will try to force the contact end through the PCB.
9. Extract the tool. Check that the new contacts are in proper position and the solder ends extend through the PCB on the opposite side.
10. Solder the contacts. Replace the spring, plunger, and keycap. Check the key for proper operation.

Keyboard Repair Parts

Description	Manufacturer's Part Number
Standard Plunger	171-40103
Solid Contact	173-30052-2
Split Contact	173-30053-2
2 oz. Spring	173-10012-4
3 oz. Spring	173-10012-2
Metal Insertion Tool	773-10000
Plastic Insertion Tool	773-10023

Interconnect Cable

When replacing the Interconnect Cable, be sure that you install the cable correctly. Insert the contacts from the top side of the Board. Bend the contacts 90° to fit the holes in the two Boards. Be sure that the contact crimps are facing the Boards.

CAUTION

With the contact crimps facing up, stress is applied to the plastic Interconnect Cable and the contacts when the Board is installed in the plastic case — so exercise extreme caution when re-installing the Boards.

Video Divider Chain

Problems in the Video Divider Chain will usually be associated with the stability of the Display. Loss of vertical or horizontal reference frequencies can sometimes be traced back to defective counters or bad Reset gates. Since the system master clock/oscillator is included in this section, inactive (dead) system troubleshooting can end up here.

Since most of the reference and timing signals for the video processor are generated in the Divider Chain, most (not all) display difficulties can be isolated to this section. This is especially true of vertical roll or horizontal tear of the display. If the horizontal or vertical reference frequency is not getting to the sync processors, then the problem definitely is a Divider Chain screw up.

The major test point to check in the chain is the 6Ø Hz output at Z32, pin 11, using a frequency counter. If 6Ø Hz is found at this pin, then the chain is probably working correctly. If not, move up the chain, toward the oscillator, until a correct signal is found.

SIGNAL NAME	SIGNAL SOURCE	SIGNAL FREQUENCY	DIVISION RATIO	COMMENTS
UDRV	Z32 Pin 11	60.00 Hz	177.3517×10^3	Vertical Reference
R8	Z32 Pin 8	60.00 Hz	177.3517×10^3	Character "Row"
R4	Z32 Pin 9	180.0 Hz	59.11722×10^3	Character "Row"
R2	Z32 Pin 12	360.0 Hz	29.55861×10^3	Character "Row"
R1	Z65 Pin 12	660.0 Hz	16.12288×10^3	Character "Row"
L8	Z12 Pin 11	1.319 kHz	8.067550×10^3	Character/Scan Line
L4	Z12 Pin 8	2.639 kHz	4.032247×10^3	Character/Scan Line
L2	Z12 Pin 9	3.959 kHz	2.687825×10^3	Character/Scan Line
L1	Z12 Pin 12	7.917 kHz	1.344082×10^3	Character/Scan Line
HDRV	Z50 Pin 11	11.835 kHz	671.9987	Horizontal Reference
C32	Z50 Pin 8	31.670 kHz	335.9993	Character Column
C16	Z50 Pin 9	63.340 kHz	167.9997	Character Column
C8	Z50 Pin 12	110.840 kHz	96.00414	Character Column
C4	Z65 Pin 8	221.69 kHz	47.99991	Character Column
C2	Z65 Pin 9	443.38 kHz	23.99995	Character Column
C1	Z43 Pin 7	886.756 kHz	12.0003	NOTE 2
Chain	Z43 Pin 9	886.756 kHz	12.0003	Divider Chain Input

NOTES

1. All Frequencies and Division Ratio calculated using 10.641099 MHz (Master CLK Frequency [−0.03% error]).

2. Signal Frequency shown is in 64 character format. Will be held low in 32 character format.

FIGURE 18. Table of Signal Frequencies for Divider Chain

Figure 18 shows a chart of frequencies that can be expected at each chain output. Slight deviations from the frequencies shown should be expected. The chart was drawn up using standard production-type units that tend to run slow by Ø.Ø3%. The division ratio column shows the factor the master oscillator needs to be divided by, to calculate the signal frequency. For example: Assume that you measure 1Ø.64ØØ MHz at pin 6 of Z42. Then the frequency of L1 is 1Ø.64ØØ MHz divided by $1.344ØØ2 \times 10^3$ which equals 7916.93 Hz. It's just one Hertz away from the frequency listed for the chart's reference frequency. Expect small deviations, but large errors usually indicate a part failure.

If you do find a large error in frequency, move down the chart until you find a correct signal. Do not automatically assume that you have found the bad part at this point. Let's say that you are missing all signals past HDRV, but C32 is good. True, Z5Ø could be bad, but so could Z66, Z49 and Z6. The best way to truly isolate the bad part without blindly replacing chips is to cut runs. First, open the run from Z5Ø, pin 11. Retest pin 11. Is it active now? If not, replace Z5Ø. If pin 11 is active, repair your etch cut carefully and cut the etch at pin 5 of Z66. Retest pin 11. If it's now active, replace Z66: if not, repair your cut and continue the cut/test process until you have isolated the shorted run.

Video RAM

If you suspect Video RAM problems, you should try a SCOATS loading. SCOATS will be most helpful in rooting out bit-error in the RAM's. If the test generates large amounts of bit-errors, you should suspect either the Divider Chain or the Video RAM addressing multiplexer. Multiple "Ready", ">" and characters all point to RAM addressing errors.

Normally, addressing errors occur when there is a short or open between the multiplexer and the RAMs. Signal activity on the address inputs of the RAMs can be easily checked with an oscilloscope. All address lines (VØ through V9) should be active in 64 character format. There will not normally be any floating conditions on these inputs. The VWR* input to Video RAM will only be active during a CPU data transfer. Normally, it should be high.

Addressing errors may also be rooted out by inspection of the display. Pull the BASIC ROM and turn on power. The screen should show @9's. If you get some @9's, examine the display carefully. If there are @9's missing in horizontal rows, then check the address inputs associated with row data (V6 through V9). If there are @9's missing in vertical columns, examine the status of the address associated with column data (VØ through V5).

When the Display is flickering (as it does in @9 mode) the CPU is constantly interrupting the Divider Chain's control over Video RAM addressing. If you are looking at RAM addresses during this time, you will see the CPU's address flickering inside the Divider Chain's address. This is normal; as a matter of fact, it is abnormal if there does not appear to be two signals on the display. The lack of the address flicker could mean the multiplexer is not working correctly, or the CPU address is not getting to the multiplexer.

If you suspect a multiplexer is not switching properly, test it. First make sure the address line going to the multiplexer is OK. Then monitor the suspected output pin of the multiplexer and ground pin 1. The output should switch from the divider chain signal to the CPU address signal. If it doesn't, you've got a bad multiplexer — replace it. (This assumes, of course, that VID* is not held low all the time because of some other problem. It is **not** recommended that a logical high be forced on VID* in cases like this. You could damage Z36 in the address decoder section.)

Video Processing

Problems in the Video Processing section can range anywhere from a blank screen to missing dots. Usually, the fault is easily found because this section is a serial-type. For example, if you have graphics problems, you know there are only two chips that are used as graphics handling devices. You would immediately look around shift register Z11 and graphics generator Z8. The parts that are strictly alphanumeric are character generator Z29 and its shift register Z1Ø. Defective devices that can affect both alphanumeric and graphics are: Z26, Z27, Z3Ø and the video mixing circuit, consisting of Q1 and Q2.

The worst video problem you can possibly get is the blank screen. Where do you start?

First, test the power supplies for proper operation. Then check the master oscillator for operation. If they are OK, move down to the emitter of Q1 and look for video. Work your way backwards until you find signal activity. Before going too far, you might check to see if the character generator is receiving data. If so, you have the problem trapped between two points. Try to determine what the system wants to do. Is it trying to output alphanumerics? Look for activity at Z26, pin 8. If this point is active and pin 6 is always high, you've just eliminated some logic. Why isn't Z1Ø outputting? If it is output-

ting, maybe Z3Ø, pin 2 is being held high for some reason. Maybe Z26, pins 8 and 6 are both high. What could cause this? Is Z27 working? Is flip-flop Z7 always reset? Maybe LATCH is not active. If so, both Z7 and Z27 will not operate.

Sometimes it helps in video troubleshooting to force a screen completely full of data. Pull BASIC ROM(s). The CPU will try to go into an @9 state. If nothing else, you will now have an easily recognizable scope pattern you can trace.

Since this section is serial, here are a few aids you can use in your troubleshooting:

Dim Display Z41 going bad. If Z41 is running hot, replace it. It's getting hot because the output transistor has increased its saturation voltage. The higher voltage does not allow the signal at the base of Q1 to swing low enough for proper video-to-sync ratio.

Missing Alphanumeric Dots If you're missing dot rows on all characters, check the line count data going to the character generator. If these lines are OK, replace the character generator. If you are missing dots in vertical columns, check dot inputs to Z1Ø. If OK, replace Z1Ø.

Unstable/Flashing Dots Sometimes Z1Ø will "miss" data on its input during a load cycle. This is usually heat associated. If the problem is cured or made worse by giving Z1Ø a shot of circuit coolant, Z1Ø must be replaced.

Missing Graphic Parts Check input to Z8. If OK, check outputs. More than likely, you'll have a broken etch around Z8. Try to determine if you are missing "right" or "left" graphics. Missing vertical cell parts are usually associated with Z8, while a graphic cell with vertical streaks indicates a problem with Z11.

Unstable/Flashing Graphics Usually harder to detect and not as common as unstable dots. But, same type of fix – replace Z11.

No Inter-Character Line Blanking Problem with L8, the frequency divider chain or latch Z27.

Severe Display Interference Usually not a Video Processor problem. Look at +5 volt supply bus. If you find oscillations, see why C12 or C13 (in 12 V supply) is not working.

Spelling Errors A system that mis-spells words usually has data screw-ups in Video RAM, or the data going to character generator Z29 is being grabbed by a short or defect around latch Z28.

Sync Generator

The Sync Generator section is one of the easiest circuits to troubleshoot. If the timing references are getting to Z6 and Z57, it is a simple process to find the point where you lose the signal. A problem can occur with the adjust pots, R2Ø or R21. Severe heat will cause these parts to fail. C2Ø and C26 are usually dependable unless they are physically crushed. You may find C21 or C27 shorted. These capacitors are mylar and are very susceptible to shorting out under impact stress.

An important point about this circuit: Z6 and Z57 are CMOS devices. Unlike TTL, they are high impedance devices that consume little current. A floating condition on a CMOS input will not necessarily give a floating "display" on an oscilloscope. A floating condition may look high or low depending on the charge of the broken line tied to the input. Even the resistance of your finger across a broken run can complete the circuit and cause a CMOS device to operate. When you remove your finger from the run or pin, circuit operation may fade away very slowly as the PCB run discharges.

Address Decoder
Expanded Discussion

Since the Address Decoder is made up of gates, it is extremely easy to fix once you find the problem. The hard part is knowing when to suspect a fault with the decoder section. Section Isolation demands that the Address Decoder be functional, at least partially. Unfortunately, there is no "cut and try" method to determine if this section is working correctly. Of course, you can monitor each output to see if it's responding, but you really can't be sure the signal is supposed to be there when it is.

For example, you see ROM* and MEM* operate, but you are not really sure if the address specified ROM*. There are only two ways to be sure ROM* is supposed to be outputted, and they are:

1. Look at all address lines going into the decoder and decode them yourself, or
2. Force the CPU into a known ROM* loop using machine language or static address switching.

However, neither of these two suggestions look promising.

Usually, an Address Decoder defect will disable one of the other sections. If you get a "Ready" upon power up but no Keyboard activity, KYBD* is easily checked for activity. Here we assume the decoder is bad at KYBD*: but there is no major problem with other sections because we do get the "Ready".

The same assumption also applies whenever we power up a computer that does not give a "Ready"; but gives us, instead, a recognizable pattern to use during Section Isolation. For example, the large colon display. You know how to generate the large colon display – pull DIP shunt Z71. Since you have large colons, with Z71 installed, what's wrong with the DIP shunt? If it appears OK, find out why RAM* is not active. Maybe DIP shunt Z3 is not working.

One important gate to consider when working with the Address Decoder is Z73, pin 6. If you do not get activity at pin 6, the whole decoder is going to be screwed up. Z73, pin 6 enables Z21, the device monitoring A12 through A14. If Z21 is never turned on, the Address Decoder will be absolutely dead. Z73, pin 6 should be the first signal you check in this section. Usually a lack of RAS* at pin 1 will kill Z73 (take that with a teaspoon of salt. If the CPU is completely lost, there's no telling what A15 will be doing. As the symbols on Z73 show, both A15 **and** RAS* must be low for a low output).

While you're bouncing around in the input section of the decoder, you should check address line activity. If you're fighting a lost CPU, you might find some signals that appear to be one state or the other. The steady state of an address line may try to lead you away from the Address Decoder, so you head toward the CPU, chasing a problem that exists only in the CPU's confusion. Try pressing the Reset button while monitoring the steady-state address line signal. You should see a few pulses flow past the probe. At least you know *something* is coming out and you're not sidetracked.

The Reset switch is a good test source to use when looking at decoder outputs too. During reset, the CPU is supposed to become quite busy. It must check ROM; stuff data in RAM; clear video and monitor the keyboard. Press the Reset switch. Something should be outputted at all decoder outputs – at least momentarily. If you are still faced with a steady state high at a decoder output, maybe the ROM is never getting sourced for the reset routine.

Watch out when checking Z21's outputs without DIP shunt Z3 installed! Remember, Z21 is an open-collector decoder device. Without Z3, the pull-up voltage is not available. You might not see anything on one of these pins unless you pull the pin up to V_{CC}.

Cassette Problems

Most of the difficulties you'll find in this section will be related to no recorder motor control. Usually, K1 (a reed relay) will have gone bad because of overwork.

Relay damage is particularly susceptible in Level II units. That buzz you hear every time you power up a Level II unit is the relay going bananas during the CPU's lengthy initialization routine. The power-up routine and the added recorder usage, because of more efficient file storage routines, make K1 earn its living.

The contacts could remain closed due to contamination, or the relay coil could open or short. The relay is easily replaced, but watch out for Murphy's law. It is very easy to install K1 on the Board backwards. Be sure you match the index mark on the Relay with the silk-screen mark on the PCB (when backwards, Z41, pin 3 sees a short to V_{CC}. It tends to make Z41 get hot — fast). Damage to Z41's relay side will probably kill the video side. Suspect a shorted relay (or shorted diode, CR3) if the display slowly dims and fades away.

If you suspect K1 is stuck, thump the relay body with your finger. That should free the contacts, and the recorder will stop. Don't pat yourself on the back for a job well done, however, until you replace that Relay. It may stick again on the next CLOAD instruction. Replace K1 and save yourself some grief later on.

Cassette Audio Processing

If you think you are having a cassette audio processing problem, you can check for activity at Z4, pin 1Ø. Try loading a long program into the Computer while monitoring pin 1Ø. It should be normally high, and go low on the audio pulses. So long as the pin is active and there are at least two pulses each bit time, you should be OK in giving Z4 a clean bill of health.

But, beware of a normally low signal at pin 1Ø of Z4! If you have active rectifier or level detector problems, pin 1Ø of Z4 can operate backwards.The output will be normally low and go high on tone pulses from the tape. The problem is usually associated with a shorted diode (CR4 or CR5), or the level detector may not have charged up capacitor C39 due to a failure of CR6, CR7 or C39. Consult the theory section for waveforms and operation of this circuit.

If you are getting good data to set-reset flip-flop Z24 during a CLOAD fault, you may have a digital problem. Try to CLOAD a long program (like Blackjack) and see if Z24, pin 8 is active. Also check activity at Z44, pin 15. You'd better hope one of these signals is not right, or you've got a long day ahead of you.

During a CLOAD, OUT* resets flip-flop Z24. If Z24 stays set, suspect Z25, pin B's gate — or follow OUT* back to its source. If INSIG* is messed up, examine Z25, pin 6 and follow IN* back to its source. If both of these signals look strange, check out port decoder Z54, Z52 and Z36, which combine to form FF*.

If all signals and Z44 look OK, suspect a RAM, CPU or ROM problem in that order. It would pay if you are familiar with the timing of INSIG* and OUTSIG* in a CLOAD condition using a known good TRS-80. It would definitely help you in Section Isolation if you find yourself considering a ROM, RAM or CPU defect.

A RAM bit error usually shows up in CLOAD when you find part of the loaded program correct and the other part garbage. Use SCOATS to dig out this type of error or, play the RAM swap game until you find the defective part.

A CPU problem is usually more common than a ROM problem in a CSAVE condition. This is true simply because you will be more concerned in getting a "Ready" on the screen than you are in seeing if the unit will CLOAD. If the CPU checks out good, you might re-examine the IN* and OUT* signals before considering ROM failure.

CSAVE problems usually point to software (ROM/RAM) or latch (Z59) difficulties. If your unit CLOAD's OK, try to CSAVE a program without using a tape in the Recorder. You can monitor the status of pin 5 of J3 for the output audio waveform. If you lack this waveform, check the status of R53 through R56 and OUTSIG* at pin 9 of Z59. Since DØ and D1 are the data lines used by Z59 during CSAVE, you might look at these two lines. Also check for a high at pin 1. You could have a solder short pulling pin 1 toward ground.

What about a 32 character display format that won't go away? Z59 is handling this function in conjunction with the status of D3. OUTSIG* clocks the latch. If OUTSIG* works fine during CSAVE and CLOAD, suspect a defective latch. Also, Z59, pin 14 could be shorted to ground. Try to clear latch Z59 by shorting pin 1 to ground for a second or two. If the display changes from 32 character to 64 character and stays that way, suspect a ROM or CPU software glitch. If the display goes from 32 character to 64 character, then back to 32 character format each time you short and release, you should suspect a defective OUTSIG* line. For some reason, OUTSIG* must be active all the time, or noise is triggering Z59 due to an open etch.

Power Supply

Most of the problems that result in loss of power supply operation will be associated with solder shorts, component shorts or bad power supply adapters. Normally, the power supply will not be damaged due to a short because the regulators use current-limiting with fold-back. A solder short or shorted component does not have to be located in the power section to cause a supply problem. The short could be anywhere.

If you are missing +12 volts and +5 volts, measure the voltage across R18. This resistor monitors the current flow from the +12-volt supply. If the voltage reads Ø.6 volts or so, the +12-volt bus is in fold-back and has shut itself off. Since the +12 volt bus is shut off, you will not have +5 volts because the +5 volt regulator is referenced to the +12-volt output. You will have to find and remove the short on the +12 volt bus before anything will work.

If you find that you are missing the —5 volt supply, first confirm that there is ample negative voltage on the adapter side of R19. See if R19 is dropping all of the voltage. If so, you have a —5 volt bus short (this assumes that CR2 has not been put in the Board backwards).

The +12 volt and the —5 volt supplies are used by System RAM. If you have problems with either of these two, suspect a RAM short. See if you can find a RAM that generates more heat than the others.

DANGER

Do not get in the habit of checking RAM temperatures with a finger! A supply short in a RAM chip can heat it up to soldering temperatures. It is quite painful to discover the RAM manufacturer's logo burned into your fingertips!

Pull all RAMs and retest. If all of the power supplies are now OK, turn off the power and reinstall a RAM. Turn on the power and retest. In-

stall each RAM until you find one that crashes the power supply. Remove the bad RAM and continue to check the rest of them. There may be more than one shorted device.

A short on the +5 volt bus can be a real headache. Unless you can see the short, you will have to cut runs to Section Isolate. Once you isolate the shorted section, you will probably still have to make other cuts to get down to the short. Do **not forget to repair your cuts.** And remember, these runs must carry considerable current. Use stranded or solid wire (22 gauge or larger) to repair cuts. (Never solder-bridge an etch cut; simple Board stress may open the solder-bridge or tear small runs loose from the Board.)

If you find a dead +12 volt bus, examine Q1's heat sink. The hardware holding the transistor/sink sandwich together may have loosened. The heat sink may have turned around and shorted against Q1's base or emitter lead. Depending on the force, it might have sheared one or more leads from the PCB. If you find a loose heat sink, shorted or not, retighten it.

Problems with the AC adapter are usually terminal. Either the fuse link in the primary winding has opened, or the wires were destroyed at the male DIN plug. If you find the fuse link blown, check rectifier package CR8 for shorted diodes. In any case replace the adapter.

Figure 19 lists the voltages found around Z1 and Z2 for a normal operating unit. The +12 volt supply has been adjusted for 12.ØØ volts and the −5 volt supply has been adjusted for 5.ØØ volts.

Z1		Z2	
Pin Number	Voltage	Pin Number	Voltage
1	0.00	1	0.00
2	5.30	2	10.60
3	5.00	3	11.99
4	5.00	4	6.92
5	5.00	5	6.92
6	7.46	6	6.92
7	0.00	7	0.00
8	0.00	8	0.00
9	0.33	9	5.72
10	5.89	10	12.31
11	11.99	11	21.16
12	11.99	12	21.69
13	7.05	13	13.48
14	0.00	14	0.00

All voltages are measured with a digital voltmeter. Voltages are referenced to ground at the right side of capacitor C9.

FIGURE 19. Table of Power Supply Voltages for Z1 and Z2

Horizontal and Vertical Adjustment

After components in the Sync Generator are replaced or after other repair work, the horizontal and vertical centering should be confirmed. Enter the following sample program:

```
10  CLS
20  FOR X=0 TO 127
30  SET (X,0) : SET (X,47)
40  NEXT X
50  FOR Y=0 TO 47
60  SET (0,Y) : SET (27,Y)
70  NEXT Y
80  FOR X = 62 TO 65
90  SET (X,23) : SET (X,24)
100 NEXT X
110 GO TO 110
```

You can load this program on a tape several times (for easy use). It will run on both Level I and Level II machines.

The re-centering program draws a large graphics rectangle on the outside boundaries of the cell array. It also draws a center rectangle. Adjust R20 and R21 so that there are equal boundaries on all sides of the large rectangle. Use a non-metallic screwdriver so body capacitance does not interfere with your adjustments.

It Doesn't Work . . . Sometimes

There is a well known rule of Murphy's Law that states: "A device will function properly whenever the operator is in a position to correct a malfunction".

There may come a time when you find your TRS-80 operates just great when it comes near a test bench. But when the moon is just right and the Computer is away from the schematics and test equipment, the memory listing looks like some unknown language and the Relay keeps time with the garbage on the display.

The only logical choice you have is to burn it in. Set up the computer in some unused corner and run the memory part of SCQATS. Hopefully, after a few hours, SCQATS will root out a spastic memory location. Try to keep the unit in the case for maximum heat retention. If all else fails, replace the RAMs as a set. Maybe replacing an entire set of RAMs will be less costly on your nerves than fighting a losing war with one bit.

If the problem appears to be in some other section where SCQATS is not effective, try to generate a program that will cause a constant failure. The only chance you have to fix an erratic problem is a faithful duplication of the malfunction. If the problem is: **For/loop statements crash part of the video section**, then use For/loop commands in your program.

Maybe there is a cold solder joint that opens every once in a while. Expect one of these when you find a problem that exists only if you flex the Board a certain way. See if you can localize the bad joint by tapping the Board with a non-conductive rod. Maybe you have a solder ball rolling around. Sometimes you can jar it out from under a socket by tapping the Board. If you're lucky, the ball may lodge somewhere and stay. Once you've got the problem to stay still, you've almost got it fixed.

Look on the etch side of the Board under the RAMs. You might find an installation error. A socket lead is easily bent. Maybe the lead was folded under the socket body and not soldered in a hole. There may be just enough pressure on the lead and the pad to allow spastic operation. Look for a smooth coating of solder instead of the pointed cone you would expect if a wire or lead was protruding through the Board. Don't limit your visual inspection to socketed parts only. It is possible to find a bent lead on a standard package, although it is not as common as a bent socket lead.

Another type of defect that can cause intermittent failure is the open feedthrough. See if you can find any solder-filled holes where the solder did not come all the way through the Board (the plating may be cracked inside the hole and the solder did not flow past the crack). Solder a small piece of lead wire through the suspect hole during repair. Defective feedthroughs usually occur in groups. If you fix one hole, look carefully at surrounding holes for other defects.

Hints, Ideas, Suggestions And . . . *What Now?*

One of the advantages of digital computers is the repetition of basic circuits. Once you know how a basic logic cell works, it will operate in the same way all through the machine, no matter how many times it is repeated. The gate may not generate the same type of signal that another one did two inches away, but the output does respond in the same way under the same input conditions.

One of the disadvantages of troubleshooting digital computers is the machine language software. There is some kind of program inside there, pushing the buttons and pulling the levers. Nobody has bothered to tell you exactly how it performs a task. You know it does exist though, like migraine potential, ready to give you a headache and a half. In the Circuit Description Section we briefly told you how the keyboard scan software operates, but the discussion never does get down to the nuts and bolts. So what do you do if you think the problem exists in software and the only way you can confirm your suspicions is to know exactly what software is doing? Before you can examine exactly what software is doing in the Keyboard, you'll have to get yourself an expensive logic analyzer, a ream of computer print outs, and spend a lotta time figuring out how the program operates. Unless you're willing to spend the time and money, software listings are not going to help in troubleshooting.

This section of the book will give you hints and ideas to try when you're faced with software-type problems that screw up the hardware. The hints will play the advantages against the disadvantages of digital troubleshooting and help you confirm or deny suspicions about where the problem really resides.

The Gate With The Static Output

How many times have you come across a logic element that has data screaming into it and an output that just sits there? Probably too many times, especially in a major area like the address decoder where you really don't need that kind of "hassle". Normally, you attach a scope to the gate inputs and see if output conditions are ever met. If you have two gate inputs and two scope channels, you'll do OK. But, you can get gray hair trying to analyze an 8-input NAND gate that way.

One useful technique you might try is the input short. When confronted with an unresponsive OR gate or NOR gate, short an input to ground. Normally, a ground short will not harm a TTL output. If you have a 2-input NOR gate, for example, you could short one input to ground and, if the gate is working properly, you should see the unshorted active signal pass through the gate. Move the short to the other input to see if that pin makes the output respond. If so, chances are you've got a good gate — try troubleshooting backwards.

When you need to check the outputs of a NAND or an AND gate, shorting inputs to ground is not going to help (unless the outputs are high for an AND gate and low for a NAND gate all the time). **Do not attempt a short to 5 volts. A short to 5 volts can damage a TTL output!** If possible, find a gate further up the line that will respond to a ground short and will cause one of the gate inputs to go high. An example of this condition can be found on the Schematic in the Video Processor section. Z9, pin 4 supplies an inverted Latch signal to Z26. If you want pins 13 or 5 of Z26 to go high, short Z9 pin 3 to ground. You can also short the inputs to Latch Z27 and cause some highs on pins 12 or 4 of Z26.

If you tried the ground-short technique and still did not get output activity, what then? Well, assuming that you met all input requirements, you have two choices:

1. the gate is truly bad or
2. there is a ground or V_{CC} short to that output.

To check for both quickly, cut the etch run (if possible) and free the output pin, then retest the gate. If you get output activity now, you have a run short. If not, replace that package — it's bad. (Don't forget to repair your etch cut!)

If you suspect a short, analyze the voltage level of the gate output. A short to V_{CC} will show about 5 volts. A normal TTL output gives a level of about 3.7 volts. (This applies to TTL only. CMOS outputs swing millivolts from the supply voltage.) If you suspect a 5-volt bus short, follow the run to its terminal point. Carefully examine places where the run gets close to a 5-volt bus.

A ground short or a logical low short may be isolated the same way. Follow the run. A ground short may also be analyzed using an oscilloscope. Hook up the scope to the node that you think is shorted. Turn off the system power while watching the scope trace. If the trace does not move in the vertical position, the short is to ground. If the trace moves up, then down, when power is removed, the run may be shorted to another TTL output. The scope method is also useful in determining if the output transistors in a gate are active.

Types Of Shorts

There are five common shorts. They are as follows:

1. The solder splash short
2. The solder ball short
3. The solder hair short
4. The etch short
5. The defective component short.

The solder splash short is probably the most common. This short develops due to excess solder and/or careless repair techniques. A solder bridge can develop between two pins on an integrated circuit during installation because of excess solder or too large a soldering iron tip. The true solder splash results when a soldering iron loses a bubble of solder, and the TRS-80 is on the receiving end. Usually, a splash is easily detected. They are big and cold-soldered to several runs.

The solder ball short usually develops between the time the factory builds the computer and the time a customer picks it up. The birth of a solder ball is at the factory's wave solder machine. During soldering, hot gasses will expand and blow liquid solder into the air over the Board. The air partially cools the solder and it sprinkles down on the Computer, with the solder already formed into little spheres of varying sizes. The balls stick to the Board because of the moist flux. The cleaning process cannot break all balls loose, so a few stay on the Board. If the balls are small enough or hidden, they may not be detected unless they cause a problem during factory testing. During packing and shipping, the solder ball may break loose and roll around until it becomes lodged under a socket or wedged between runs. So you end up with a dead machine and some nasty thoughts about the factory.

The solder hair is an extremely fine sliver of solder that can short unprotected runs. Solder hairs usually develop when a solder coated etch is rubbed with a sharp tool or an abrasive material. Impact stress causes the coating of solder to surface splinter. The splinters are then dragged across runs by hand or by vibration. If you cut runs when trouble shooting, take care that you do not over cut. If you do, re-flow solder cut marks you accidentally made in the runs with the iron to melt the splinters. Never. NEVER. **NEVER try to clean solder points or paths on the TRS-80 with steel wool or a sharp tool!** A rubber eraser, used with light pressure, is all that is necessary if any cleaning is needed.

An etch short means "incomplete copper removal between circuit runs during Board manufacturing". The etch short in the field would show up as damage to a Board run. Heat was used to force the copper and base material to join, and heat can take the copper right off again. Excess heat during soldering or letting the soldering iron rest against a run can cause the pad, or run, to slide. The pad doesn't have to slide too far to short nearby circuit runs.

The component short is the defective gate or bad diode. There is not too much to mention here. Component shorts between a power bus and ground will usually fold back the power supply.

Logic Shorts

Two or more TTL gate outputs may become shorted together and create strange problems. Some functions may work, while others may not. Multiple shorted gate outputs can be recognized by finding a tri-state type signal where there isn't a tri-state device driving it. A tri-state signal will have true logical high and low voltage levels. There may be places along the waveform where the voltage level is between a high and a low. Look on a data bus line to see a typical tri-state wave shape. A TTL output belonging to a gate that is not tri-stateable should never have three logic levels. If you find a screwed up wave shape, follow that run, checking nearby runs as you look for another messed up signal. When you find the two bad signals, inspect the run closely for shorts. If you don't find any shorts, keep tracing until one of the runs terminates. The two runs may come together somewhere else on the Board.

The counters used in the divider chain are 74LS93's. This family of TTL ripple-counters sometimes shows a multilevel high on the data outputs. The wave shape appears to have one or more steps while the main pulse is high. This type of output is satisfactory if the steps do not fall below the 2.4 volt minimum logical high level. Usually, you will have a counter step of about half a volt or so, and the lowest step will never fall below 3.Ø volts. This is just a little tidbit that could side track a technician who is not familiar with the '93 counter.

Address and Data Line Shorts

A short in the address or data lines is about the worst problem a Computer can have. The two busses go everywhere; and it only takes a small flake of solder to kill a system as dead as shooting it would. If the two busses were rated as to which is worse to have a short, the data bus would win. Since the data bus is a two way street, everything attached to it is tri-stateable. You cannot really separate two shorted lines from the rest, because they all look strange. On the address bus, there are no tri-state devices (there are buffers used, but they are never tri-stated during normal operation).

Address line shorts are rather straightforward. Try finding two signals with tri-state type voltage levels. (The CPU address buffer is not tri-stated unless the "Test" input is grounded at the expansion connector.) After finding the two bad lines, you can follow one around the Board until you find an area where the two runs come close to each other. If you still can't find the short, try cutting one run until you've isolated the area where a short exists. Be sure you repair each cut after each check. Don't leave them for later, you might forget where you made the cuts and create even more problems.

Data line shorts need to be isolated in the same way, once you find the two bad lines. Finding the bad lines is another matter altogether. The best way to search out shorted data lines is to disable the Data Bus. Short the TEST* signal to ground. All data and address line buffers will tri-state. If you suspect a gate short to data line, look at all data lines. You are looking for one that is not floating. With a short between data lines, you will need to pull up a line to 5 volts with a 4.7K resistor and check on the other lines for a high, instead of a floating condition.

If you spend a lot of time hunting bus shorts, you might want to build yourself a little test board. The board could contain pull up resistors, LEDs and switches. You can connect this type of fixture to the expansion connector and rapidly determine if you have a bus short or not by switching ground to each line and see if any other lights go on. It's better than trying to grow a third arm so you can handle the pull up resistor, the scope probe and the shorting lead all at the same time!

Bent Pins

On socketed parts, it is easy to replace devices for troubleshooting. But, be careful that you do not cause more problems by getting in a hurry. On RAMs, it's extremely easy to bend an IC pin between the socket and the part. Suspect a bent pin if the part is hard to insert; and, upon more pressure, it suddenly snaps in. Make it part of your isolation routine to peek under the CPU chip and look for folded-under leads. The decoupling capacitors near the RAM may prevent you from looking at these sockets. However, you can usually inspect the RAM leads from the top of the Board. You can also check that the RAMs are level and of uniform height. Maybe a RAM has been loosened and only one side is attached. A quick push on the loose part may be all that's necessary to fix a malfunctioning unit.

DIP Shunts

It's been said before, but it needs to be said again. Be careful when programming DIP shunts. If undue stress is applied to a DIP shunt while programming, the plastic may crack. The crack usually develops in the center of the part in line with the shorting bar's narrowest point. The bars may separate enough to cause an open. When you program a DIP shunt, first install the part in the socket. Use a scribe to break the bars you need to open. Use only enough pressure to break the bars (don't try to drive the scribe through the Board). Be careful when you break bars near pins 1 and 8 as this is the weakest part of the shunt.

If you suspect a cracked shunt, check resistance of the unbroken bars. Do not press hard with the meter probes. You might temporarily close a cracked bar and it'll read OK. If you find a defective bar, replace the shunt. Solder a broken shunt only if a replacement is not available.

The Outside World

TRS-80 and The Outside World

As you use your Computer more and more, you will begin to see more and more applications for it. There will come a time when you will want to do some tasks that can only be handled with external circuitry; or the BASIC language does not include such functions as "TURNCP.X" (turn on coffee pot at time x). You know that a Computer should be able to do such things but you may not know how to go about designing a circuit to do it.

An added function consists of two basic parts; the first part is the software. Will you have to write a short machine language program to do the task? If you have a Level I machine, more than likely the answer is yes. If you have a Level II machine then maybe you can use the POKE or PEEK or the OUT or IN instructions. If so, you're writing the program in BASIC and the software part will not give you any problems. Software written in machine language can also be painless if you use an Editor/Assembler to generate your object code. At any rate you're going to need software to control the device that you build.

The next part you'll need to think about is the hardware. That's a necessity. Neither the TRS-80 nor the Expansion unit incorporate a relay to turn on power to your coffee pot (or whatever). Hooking up a relay to an output line isn't going to do the job, because your coffee pot may turn on every time you do a cassette load.

When designing hardware for your special task, you should have defined the software already. How you design your hardware will be determined by the instructions you will use to operate it. There are two approaches you may use to control your home-built device:

1. A memory mapped system
2. A port system

If you decide to memory map your hardware, you will specify a memory address that will be the location your system is in the map. To write data to your system you will address it via the 16 address lines and write data to it using the data lines. You will do this by using the address lines and the WR* line at the Expansion Port Connector. If you need to read data back into the TRS-80, you will use the address lines and the RD* line. The machine language software to do the job will be the LD (load) instruction or in Level II the POKE and PEEK instruction.

For example, let's say your coffee pot is memory-mapped to address 8FFF (Hex) and binary data Ø2 will turn it on. An assembler instruction that controls it will be:

```
LD 8FFFH,Ø2H ; turn on coffee pot.
```

(Address 8FFF will be loaded with binary Ø2 Hex.)

Note: This assembler routine will generate the object machine code which will activate the function.

A POKE statement would look like this:

```
POKE 36863.2
```

(decimal address 36863 is 8FFF in Hex.)

In a port-based system, you will specify a port address out of the 256 ports the CPU will address. Once again the address of your system is selected using the address lines but this time only eight (8) lines are used instead of all 16. The data bus is used to pass information back and forth between the CPU and the selected port.

For example, if your coffee pot circuit is port addressed at FE Hex and Ø2 Hex will turn it on, an assembler routine that generates the machine code to load Ø2 into port FE is as follows:

```
OUT ØFE,Ø2H ; Turn on coffee pot
```

In Level II language you could say:

```
OUT 255,2
```

(This is in BASIC and does the same thing.)

The port instruction uses OUT* and IN* and the 8 lower order address lines.

Should you memory-map or should you use ports? This is entirely up to you. If you memory map, you should select your address lines high enough so that you will not interfere with your in-system RAM. If you have maximum memory you may not be able to go high enough to be free from RAM. But if you have only 16K or less of RAM, you have thousands of potential addresses to use. If you use ports, you only have 255 available (you'll remember from our earlier discussion that port FF is used for the cassette recorder).

In terms of hardware, using ports will be slightly better than memory mapping. Why? Address decoding. In a port-based system you will only need to decode 1 out of 256 possibilities. Only 8 lines. In a memory-mapped system you will have to decode 1 out of 65,536 possibilities — that's 16 address lines. We'll show it both ways and let you make up your own mind.

Memory Mapped External Device

Figure 20 shows a logic diagram of a memory-mapped coffee pot switch (of course it could be any other control switching circuit). It was drawn to illustrate a technique and should not be considered a final working design.

The following software requirements were assumed:

1. Needs to control a relay that will control the power going to the coffee pot.
2. Memory mapped at Hex address 8FFF.
3. Design should include some method to test if relay is on or off.
4. A program to control this external device is shown below.

At line 600 in the program, the CPU will POKE data 2 (Hex 02) into memory-location 4092 (Hex 8FFF). In binary, 8FFF is 1000 1111 1111 1111.

This 16-bit word needs to be decoded. Z1, Z2A and Z3 decode the address lines, producing 8FFF* at the output of Z3. This signal is applied to the input of Z2 C and D. The POKE statement will cause WR* to go low at the same time as 8FFF* goes low. Therefore the output at Z2 C will go high. This signal is called the WRITE and is applied to Z6, a latch. 02 in Hex is 0000 0010 in binary. Therefore, at the same time that WRITE goes high, Data bus bit D1 will be high. The high at D1 will be latched or stored in Z6. The resulting high at the Q output of Z6 will be passed thru relay driver Z7, and the relay will close and stay closed. One set of K1's contacts is used to turn on the coffee pot. The other contact is used in an acknowledge feature.

At line 800 of the program, the coffee pot circuit is once again accessed. Address decoding

```
 100 REM ***COFFEE POT POWER CONTROL***
 200 CLS : PRINT @512, "DO YOU WANT THE COFFEE POT TURNED ON";
 300 INPUT A$: IF A$ = "NO" GOTO 7840
 400 REM ***IF NO, BRANCH TO "RUNNING LATE" ROUTINE***
 500 REM ***ANYTHING ELSE, TURN ON COFFEE POT RELAY***
 600 POKE -4092,2
 700 REM ***NOW TEST IF CONTROL RELAY CHANGED STATES***
 800 B=2: A=PEEK -4096
 900 IF A AND B = 0 THEN GOTO 1980 ELSE GOTO 3744
1000 REM ***IF RELAY WORKED, BRANCH TO "WEATHER SENSOR" ROUTINE***
1100 REM ***IF RELAY DID NOT WORK, BRANCH TO "SYSTEM FAULT ISOLATE" ***
```

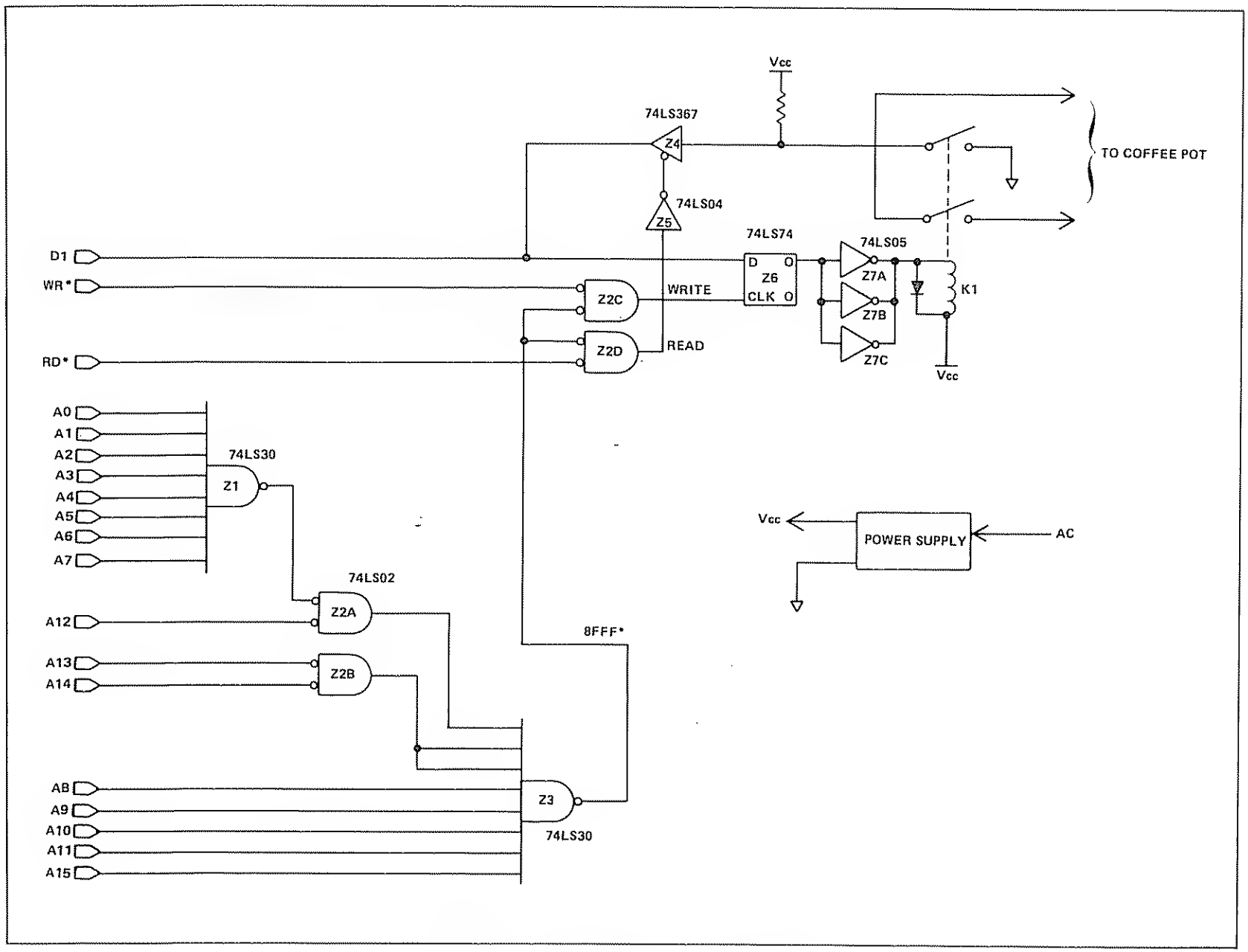


FIGURE 20. Memory Mapped Coffee Pot Control

remains the same but this time Z2D is active. Because, instead of WR* going low, RD* goes low and READ is generated. READ is inverted by Z5 and activates tri-state buffer Z4. Data present on the input is transferred to the output. If the relay is "on", Z4 will output to D1 a low; and if K1 is not on, a high will be outputed to D1. The program will take appropriate action depending on a status of D1 as line 900 shows.

Notice that the hardware:

1. Decoded the addresses and outputed one signal only when the device was called upon to work

2. Had to perform a WRITE function as well as a READ function

3. Had to latch data from the data lines (may not always be necessary)

4. Contains a separate power supply

5. Does not contain more than 1 LS TTL load on any one output from the Computer

4 and 5 are very important if you want to guarantee proper operation of your Computer.

Port Based External Device

Figure 21 shows a logic diagram of a port-based coffee pot control system. Notice from Z2 to the right, the circuit is the same as Figure 20. Only the reference designators have changed. The major difference is how a port is decoded versus memory decoding. A port is restricted to only 256 lines. Therefore only one NAND gate and part of an inverter are necessary to decode port FE (Decimal 254).

In the program listing, Lines 600, 800 will have to be changed. Line 600 should now read

```
OUT 254,2
```

And Line 800 would read

```
B=2: A=INP 254
```

When Line 600 is executed, Binary 1111 1110 will be outputed on A7 thru A0. The outputs of Z1 generates FE*. FE* is applied to Z2A and B. OUT* will go low and D1 will go high. The high at the D input of Z5 will be latched in, and the relay will turn on.

At the new line 800, FE* will be generated and IN* will go low. Z2B generates READ, and the status of the contacts of K1 will be sent to the Computer.

As you can see, the major differences between a memory-mapped system and a port system is the decoding techniques. Even though different signals are used, they operate in about the same manner.

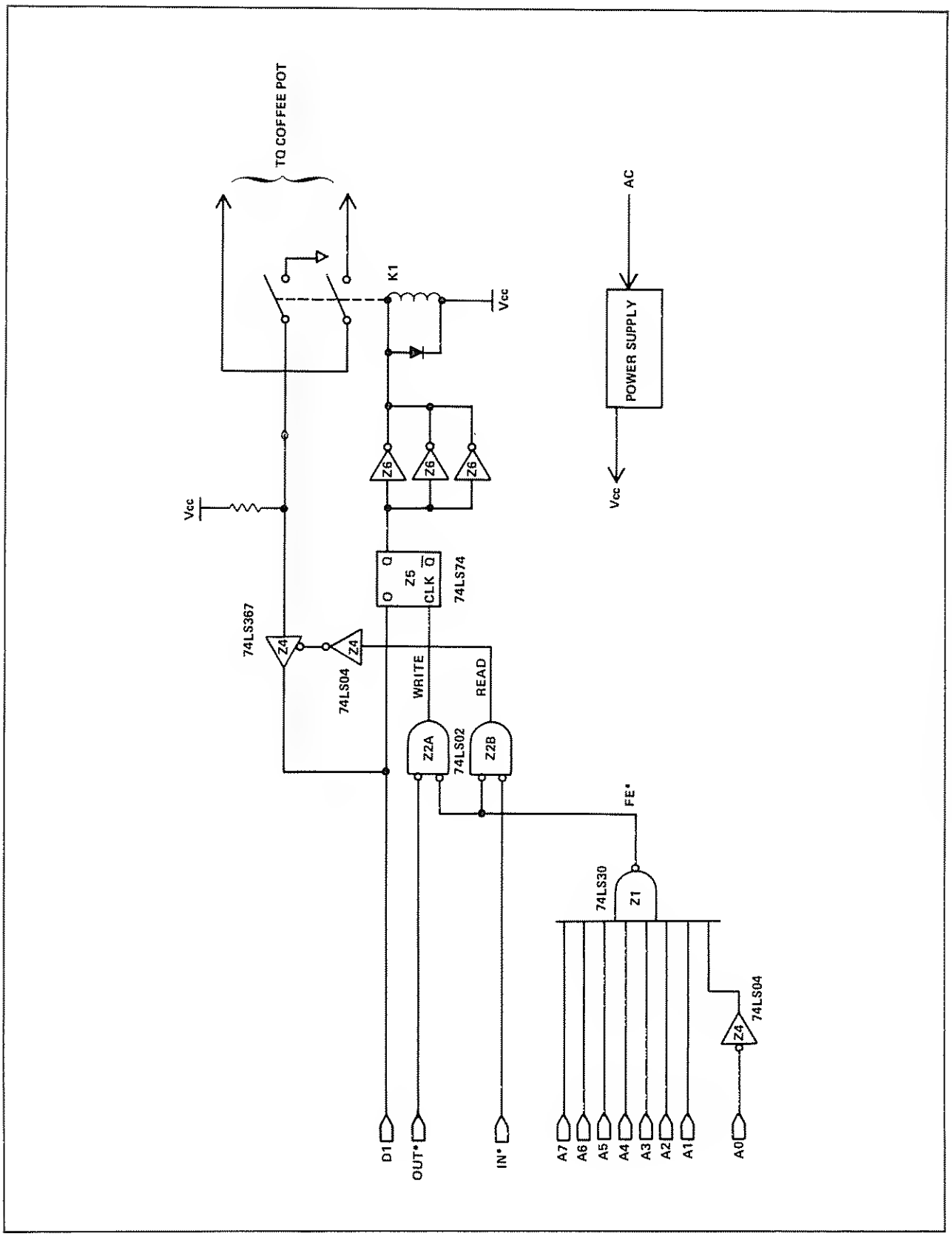


FIGURE 21. Port Based Coffee Pot Control

Explanation of Expansion Port Signals

The table on the facing page lists all the Expansion Port pin connections and signal names. Figure 22 shows the connection points as they exist on the back of the Logic PCB. The following detailed description of the various signals will aid you in understanding (and using) the Expansion Port.

Address Output: There are 16 of these lines, labeled AØ thru A15. AØ is the least significant bit, and A15 is the most significant bit. These outputs form the address bus from the TRS-80 microprocessor. If you monitor these lines, you would know exactly where in memory the CPU was reading or writing data. The address bus can address up to 65,536 different memory locations (2^{16} = 65,536). Each line will drive only one TTL load.

Bi-directional Data Bus: There are 8 of these lines, labeled DØ thru D7. DØ is the least significant bit and D7 is the most significant bit. The CPU uses the data lines to move binary data from one section to another section in the Computer. Since this bus is bi-directional you should use tri-state buffers for input and output data moves.

Row Address **Strobe Output:** This line is labeled RAS*. It is normally high and goes low only when the CPU is outputting the ROW portion of the address. It is used to address dynamic RAMS. (See RAM Addressing in text for operation.)

Column Address Strobe Output: This line is labeled CAS*. It is normally high and goes low only when the CPU is outputting the column portion of an address. It too is used to address Dynamic RAMS. (See text for operation.)

Multiplexer Control Output: This output is labeled MUX. It is used to select the proper address line in conjunction with RAS* and CAS* for the RAMS. (See text for operation.)

System Reset Output: This output is labeled SYSRES*. It goes low only when the Reset button is pressed, or upon power up. It can be used to reset external devices at the same time as the TRS-80 is reset. Normally it will be high.

Test Input: This line is labeled TEST*. When taken low it will tri-state the data, the address and the control group buffers. Normally, it will not be used by external circuits. It is used only during factory testing and in some cases during troubleshooting.

Processor Wait: When taken low, WAIT* will pause the CPU from further processing until WAIT* goes back high. In some cases an external device may need time to gather data. The WAIT* input will give the external device the time it needs.

Memory Write Strobe: When WR* goes low, the CPU is writing the data present on the data bus into the memory specified by the address bus. It is normally high.

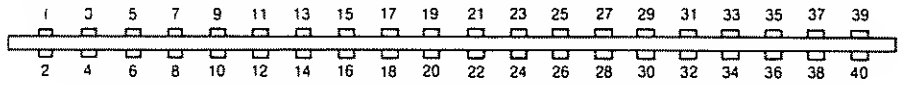
Memory Read Strobe: When RD* goes low, the CPU is reading data present on the data bus from the memory location specified by the address bus. It is normally high.

Peripheral Write Strobe: OUT* operates somewhat like WR*; except OUT* is a port function. When OUT* goes low, data present on the data bus is meant to be received by the port specified by the 8 lower order bits in the address bus, AØ thru A7. The Z8Ø can address up to 256 output ports.

Pin Connections for Expansion-Port Edge Card

P/N	SIGNAL NAME	DESCRIPTION
1	RAS*	Row Address Strobe Output for 16-Pin Dynamic Rams
2	SYSRES*	System Reset Output, Low During Power Up Initialize or Reset Depressed
3	CAS*	Column Address Strobe Output for 16-Pin Dynamic Rams
4	A1Ø	Address Output
5	A12	Address Output
6	A13	Address Output
7	A15	Address Output
8	GND	Signal Ground
9	A11	Address Output
10	A14	Address Output
11	A8	Address Output
12	OUT*	Peripheral Write Strobe Output
13	WR*	Memory Write Strobe Output
14	INTAK*	Interrupt Acknowledge Output
15	RD*	Memory Read Strobe Output
16	MUX	Multiplexor Control Output for 16-Pin Dynamic Rams
17	A9	Address Output
18	D4	Bidirectional Data Bus
19	IN*	Peripheral Read Strobe Output
20	D7	Bidirectional Data Bus
21	INT*	Interrupt Input (Maskable)
22	D1	Bidirectional Data Bus
23	TEST*	A Logic "Ø" on TEST* Input Tri-States AØ-A15, DØ-D7, WR*, RD*, IN*, OUT*, RAS*, CAS*, MUX*
24	D6	Bidirectional Data Bus
25	AØ	Address Output
26	D3	Bidirectional Data Bus
27	A1	Address Output
28	D5	Bidirectional Data Bus
29	GND	Signal Ground
30	DØ	Bidirectional Data Bus
31	A4	Address Bus
32	D2	Bidirectional Data Bus
33	WAIT*	Processor Wait Input, to Allow for Slow Memory
34	A3	Address Output
35	A5	Address Output
36	A7	Address Output
37	GND	Signal Ground
38	A6	Address Output
39	+5V	5 Volt Output (Limited Current)
40	A2	Address Output

NOTE: *means Negative (Logical "Ø") True Input or Output



Mates with AMP P/N 88103-1 Card Edge Connector or Equivalent

FIGURE 22. Connection points for Expansion-Port Edge Card (viewed from rear of keyboard assembly)

Peripheral Read Strobe: IN* operates somewhat like RD*: except IN* is a port function. When IN* goes low, the CPU is looking for data on the address bus that comes from the port specified by the 8 lower order bits of the address bus (AØ thru A7). The ZØØ will address up to 256 input ports.

Interrupt Input: INT*, when taken low, will force the CPU to a predetermined address in ROM. There are three modes of operation. In the first, this line is ignored. In the second mode, INT* causes the CPU to go to address ØØ38 Hex and to go on from there. In the last mode, the CPU may go anywhere in memory you want but this function cannot be used in the TRS-80 due to internal circuits. See a Z-8Ø technical manual for more information on this input.

Interrupt Acknowledge: INTAK* goes low whenever the CPU enters an interrupt mode. It is useful for external circuitry to know that the CPU actually did go into an interrupt routine.

System Ground: Ground is the reference point of all voltages and logic levels in the TRS-80.

5 **Volt Output:** This line comes from the 5-volt power supply. The power supply in the TRS-80 is designed to drive just the TRS-80 and a little more for overhead. It is suggested that you do not use this output for external devices. *In Level II machines this pin has been modified to show a ground, instead of 5 volts.*

Parts List

Parts List

LEVEL I Logic Board

Symbol	Description	Part Number
	Printed Circuit Board, Logic	1700069

CAPACITORS

C1	200 μF, 16V, Electrolytic, Axial	1500059
C2	10 μF, 16V, Electrolytic, Radial	1500012
C3	0.01 μF, 10%, 25V, Disc	1500047
C4	10 μF, 16V, Electrolytic, Radial	1500012
C5	10 μF, 16V, Electrolytic, Radial	1500012
C6	100 μF, 16V, Electrolytic, Radial	1500014
C7	0.01 μF, 10%, 25V, Disc	1500047
C8	2,200 μF, 35V, Electrolytic, Axial	1500064
C9	10,000 μF, 16V, Electrolytic, Axial	1500058
C10	10 μF, 16V, Electrolytic, Radial	1500012
C11	10 μF, 16V, Electrolytic, Radial	1500012
C12	470 pF, 50V, Disc	1500057
C13	470 pF, 50V, Disc	1500057
C14	0.01 μF, 10%, 25V, Disc	1500047
C15	0.01 μF, 10%, 25V, Disc	1500047
C16	0.1 μF, 10%, 12V, Disc	1500052
C17	0.1 μF, 10%, 12V, Disc	1500052
C18	0.1 μF, 10%, 12V, Disc	1500052
C19	0.1 μF, 10%, 12V, Disc	1500052
C20	330 pf, 10%, 50V, Disc	1500062
C21	750 pF, 10%, 50V, Disc	1500050
C22	0.1 μF, 10%, 12V, Disc	1500052
C23	0.1 μF, 10%, 12V, Disc	1500052
C24	220 pf, 10%, 50V, Disc	1500061
C25	220 pF, 10%, 50V, Disc	1500061
C26	0.047 μF, 100V, Polyester Film	1500051
C27	0.022 F, 100V, Polyester Film	1500023
C28	0.1 μF, 10%, 50V, Disc	1500053
C29	0.1 μF, 10%, 12V, Disc	1500052
C30	0.1 μF, 10%, 50V, Disc	1500053
C31	0.1 μF, 10%, 12V, Disc	1500052
C32	0.1 μF, 10%, 50V, Disc	1500053
C33	0.1 μF, 10%, 12V, Disc	1500052
C34	0.1 μF, 10%, 50V, Disc	1500053
C35	0.1 μF, 10%, 12V, Disc	1500052
C36	0.1 μF, 10%, 12V, Disc	1500052
C37	0.1 μF, 10%, 12V, Disc	1500052
C38	0.1 μF, 10%, 12V, Disc	1500052
C39	0.1 μF, 10%, 12V, Disc	1500052
C40	0.1 μF, 10%, 12V, Disc	1500052
C41	0.1 μF, 10%, 12V, Disc	1500052
C42	22 μF, 16V, Electrolytic, Radial	1500055
C43	47 pF, 10%, 50V, Disc	1500048
C44	0.1 μF, 10%, 12V, Disc	1500052
C45	0.1 μF, 10%, 12V, Disc	1500052
C46	0.1 μF, 10%, 12V, Disc	1500052
C47	0.1 μF, 10%, 12V, Disc	1500052
C48	0.1 μF, 10%, 12V, Disc	1500052
C49	0.1 μF, 10%, 12V, Disc	1500052
C50	0.1 μF, 10%, 12V, Disc	1500052
C51	0.1 μF, 10%, 12V, Disc	1500052
C52	0.1 μF, 10%, 12V, Disc	1500052
C53	0.1 μF, 10%, 12V, Disc	1500052
C54	0.1 μF, 10%, 12V, Disc	1500052
C55	0.1 μF, 10%, 12V, Disc	1500052
C56	0.1 μF, 10%, 12V, Disc	1500052
C57	10 μF, 16V, Electrolytic, Radial	1500012

DIODES

CR1	1N4735, 10%, 6.2V, Zener	4800021
CR2	1N5231, 5%, 5.1V, Zener	4800022
CR3	1N4148, 75V	4800002
CR4	1N4148, 75V	4800002
CR5	1N4148, 75V	4800002
CR6	1N4148, 75V	4800002
CR7	1N4148, 75V	4800002
CR8	Bridge Rectifier, MDA202, 2A, 202V	4800023
CR9	1N982, 75V, Zener	4800026
CR10	1N982, 75V, Zener	4800026

JACKS

J1	Connector, Socket, DIN, 5 Pin	2100033

Symbol	Description	Part Number
J2	Connector, Socket, DIN, 5 Pin	2100033
J3	Connector, Socket, DIN, 5 Pin	2100033
	RELAY	
K1	5V Relay	4500001
	TRANSISTORS	
Q1	2N3904, NPN	4822001
Q2	MPS3906, PNP	4822003
Q3	TIP29, Driver	4820004
Q4	2N6594, Power	4824003
Q5	MPS3906, PNP	4822003
Q6	MJE34, Power	4824002
	RESISTORS	
R1	68 ohm, 1/2W, 5%	4708022
R2	2.7 K, 1/4W, 5%	4704056
R3	750 ohm, 1/4W, 5%	4704044
R4	0.33 ohm, 2W, 5%	4717004
R5	1K Trim Pot, 30%	4750019
R6	1.2 K, 1/4W, 5%	4704049
R7	1.2 K, 1/4W, 5%	4704049
R8	100 K, 1/4W, 5%	4704087
R9	3.3 K, 1/4W, 5%	4704058
R10	1 K, Trim Pot, 30%	4750019
R11	3.3 K, 1/4W, 5%	4704058
R12	3.3 K, 1/4W, 5%	4704058
R13	2.2 K, 1/4W, 5%	4704054
R14	12 K, 1/4W, 5%	4704070
R15	1.5 K, 1/4W, 5%	4704050
R16	1.2 K, 1/4W, 5%	4704049
R17	2 K, 1/4W, 5%	4704053
R18	5.6 ohm, 3W, 5%	4717003
R19	220 ohm, 1/2W, 5%	4708032
R20	100 K, Trim Pot, 20%	4750018
R21	100 K, Trim Pot, 20%	4750018
R22	75 ohm, 1/4W, 5%	4704023
R23	120 ohm, 1/4W, 5%	4704027
R24	680 K, 1/4W, 5%	4704100
R25	1.6 Megohm, 1/4W, 5%	4704106
R26	1 Megohm, 1/4W, 5%	4704102
R27	330 ohm, 1/4W, 5%	4704036
R28	270 ohm, 1/4W, 5%	4704034
R29	1.8 K, 1/4W, 5%	4704052
R30	47 ohm, 1/4W, 5%	4704019
R31	10 ohm, 1/4W, 5%	4704011
R32	10 K, 1/4W, 5%	4704068
R33	360 K, 1/4W, 5%	4704098
R34	470 K, 1/4W, 5%	4704097
R35	470 K, 1/4W, 5%	4704097
R36	360 K, 1/4W, 5%	4704098
R37	560 K, 1/4W, 5%	4704099
R38	470 K, 1/4W, 5%	4704097
R39	4.7 K, 1/4W, 5%	4704061
R40	4.7 K, 1/4W, 5%	4704061
R41	470 K, 1/4W, 5%	4704097
R42	1.0 Megohm, 1/4W, 5%	4704102
R43	10 K, 1/4W, 5%	4704068
R44	10 K, 1/4W, 5%	4704068
R45	470 K, 1/4W, 5%	4704097
R46	910 ohm, 1/4W, 5%	4704046
R47	10 K, 1/4W, 5%	4704068
R48	4.7 K, 1/4W, 5%	4704061
R49	4.7 K, 1/4W, 5%	4704061
R50	4.7 K, 1/4W, 5%	4704061
R51	4.7 K, 1/4W, 5%	4704061
R52	910 Ohm, 1/4W, 5%	4704046
R53	1.2 K, 1/4W, 5%	4704049
R54	7.5 K, 1/4W, 5%	4704066
R55	7.5 K, 1/4W, 5%	4704066
R56	220 K, 1/4W, 5%	4704092
R57	4.7 K, 1/4W, 5%	4704061
R58	4.7 K, 1/4W, 5%	4704061
R59	4.7 K, 1/4W, 5%	4704061
R60	4.7 K, 1/4W, 5%	4704061
R61	4.7 K, 1/4W, 5%	4704061
R62	4.7 K, 1/4W, 5%	4704061
R63	4.7 K, 1/4W, 5%	4704061
R64	330 ohm, 1/4W, 5%	4704036
R65	10 K, 1/4W, 5%	4704068
R66	4.7 K, 1/4W, 5%	4704061
R67	100 ohm, 1/4W, 5%	4704025

*This part is included in Level II and is incorporated on all P.C. Boards of version G and after.

Symbol	Description	Part Number
	SWITCHES	
S1	4PDT Push	5102008
S2	DPDT Push	5102009
	SINKS	
Sink Q4 Heatsink		5300003
Sink Q6-14 Heatsink		5300002
	SOCKETS	
X3	16 Pin I.C. Socket	2100037
X13	16 Pin I.C. Socket	2100037
X14	16 Pin I.C. Socket	2100037
X15	16 Pin I.C. Socket	2100037
X16	16 Pin I.C. Socket	2100037
X17	16 Pin I.C. Socket	2100037
X18	16 Pin I.C. Socket	2100037
X19	16 Pin I.C. Socket	2100037
X20	16 Pin I.C. Socket	2100037
X32	24 Pin I.C. Socket	2100034
X33	24 Pin I.C. Socket	2100034
X39	40 Pin I.C. Socket	2100035
X71	16 Pin I.C. Socket	2100037
	CRYSTAL	
Y1	10.6445 MHz, 0.004%, Series Res.	2300004
	INTEGRATED CIRCUITS	
Z1	723, DIP, Voltage Regulator	3100001
Z2	723, DIP, Voltage Regulator	3100001
Z4	LM3900, Dual Input Norton Amp.	3100002
Z5	74C00 CMOS, Quad 2-Input NAND Gate	3102026
Z6	74C04 CMOS, Hex Inverter	3102027
Z7	74LS74, Dual D Positive-Edge-Triggered Flip-Flop with Preset and Clear	3102015
Z8	Z4LS153, Dual 4-Line to 1-Line Data Selector/Multiplexer	3102019
Z9	74LS04, Hex Inverter	3102008
Z10	74LS166, 8-Bit Parallel In/Serial Out Shift Register	3102021
Z11	74LS166, 8-Bit Parallel In/Serial Out Shift Register	3102021
Z12	74LS93, Divide by 8 Binary Counter Selector/Multiplexer	3102017
Z21	74LS156, Dual 2-Line to 4-Line Decoder/Demultiplexer	3102028
Z22	74LS367, TRI-STATE Hex Buffer	3102024
Z23	74LS32, Quad 2-Input OR Gate	3102014
Z24	74LS132, Quad 2-Input NAND Gate	3102018
Z25	74LS32, Quad 2-Input OR Gate	3102014
Z26	74LS20, Dual, 4-Input NAND Gate	3102011
Z27	74LS175, Quad D Flip-Flop with Clear	3102023
Z28	74LS174, Hex D Flip-Flop with Clear	3102022
Z29	MCM6670, Character Generator	310B001
Z30	74LS02, Quad, 2-Input NOR Gate	3102007
Z31	74LS157, Quad 2-Line to 1-Line Data Selector/Multiplexer	3102020
Z32	74LS93, Divide by 8 Binary Counter Selector/Multiplexer	3102017
Z33	2 K x 8 ROM A, 450 ns, 2 Patterns	310B011
Z34	2 K x 8 ROM B, 450 ns, 2 Patterns	3108012
Z35	74LS157, Quad 2-Line to 1-Line Data Selector/Multiplexer	3102020
Z36	74LS32, Quad 2-Input OR Gate	3102014
Z37	74LS02, Quad 2-Input NOR Gate	3102007
Z38	74LS367, TRI-STATE Hex Buffer	3102024
Z39	74LS367, TRI-STATE Hex Buffer	3102024
Z40	Z80 Microprocessor Circuit, Plastic	3110001
Z41	75452, Relay Driver	3106002
Z42	74LS04, Hex Inverter	3102008
Z43	74LS157, Quad 2-Line to 1-Line Data Selector/Multiplexer	3102020
Z44	74LS367, TRI-STATE Hex Buffer	3102024
Z45	2102, AN-4L, 1 K Static RAM	3108002
Z46	2102, AN-4L, 1 K Static RAM	3108002
Z47	2102, AN-4L, 1 K Static RAM	310B002
Z48	2102, AN-4L, 1 K Static RAM	3108002
Z49	74LS157, Quad 2-Line to 1-Line Data Selector/Multiplexer	3102020

Symbol	Description	Part Number
Integrated Circuits		
Z50	74LS93, Divide by 8 Binary Counter Selector/Multiplexer	3102017
Z51	74LS157, Quad 2-Line to 1-Line Data Selector/Multiplexer	3102020
Z52	74LS04, Hex Inverter	3102008
Z53	74LS132, Quad 2-Input NAND Gate	3102018
Z54	74LS30, Triple 3-Input NOR Gate	3102013
Z55	74LS367, TRI-STATE, Hex Buffer	3102024
Z56	74LS92, Divide by 6 Binary Counter Selector/Multiplexer	3102016
Z57	74C04 CMOS, Hex Inverter	3102027
Z58	74LS92, Divide by 6 Binary Counter Selector/Multiplexer	3102016
Z59	74LS175, Quad D Flip-Flop with Clear	3102023
Z60	74LS367, TRI-STATE Hex Buffer	3102024
Z61	2102, AN-4L, 1 K Static RAM	3108002
Z62	2102, AN-4L, 1 K Static RAM	3108002
Z63	2102, AN-4L, 1 K Static RAM	3108002
Z64	74LS157, Quad 2-Line to 1-Line Data Selector/Multiplexer	3102020
Z65	74LS93, Divide by 8 Binary Counter Selector/Multiplexer	3102017
Z66	74LS11, Triple 3-Input AND Gate	3102010
Z67	74LS367, TRI-STATE Hex Buffer	3102024
Z68	74LS367, TRI-STATE Hex Buffer	3102024
Z69	74LS74, Dual D Positive-Edge-Triggered Flip-Flop with Preset and Clear	3102015
Z70	74LS74, Dual D Positive-Edge-Triggered Flip-Flop with Preset and Clear	3102015
Z71	Not used	
Z72	74LS367, TRI-STATE Hex Buffer	3102024
Z73	74LS32, Quad 2-Input OR Gate	3102014
Z74	74LS00, Quad 2-Input NAND Gate	3102006
Z75	74LS367, TRI-STATE Hex Buffer	3102024
Z76	74LS367, TRI-STATE Hex Buffer	3102024

4K RAM Kit

Symbol	Description	Part Number
A3	DIP Shunt	2100041
A71	DIP Shunt	2100041
Z13	4096 bit, Dynamic RAM, 450 ns	3108003
Z14	4096 bit, Dynamic RAM, 450 ns	3108003
Z15	4096 bit, Dynamic RAM, 450 ns	3108003
Z16	4096 bit, Dynamic RAM, 450 ns	3108003
Z17	4096 bit, Dynamic RAM, 450 ns	3108003
Z18	4096 bit, Dynamic RAM, 450 ns	3108003
Z19	4096 bit, Dynamic RAM, 450 ns	3108003
Z20	4096 bit, Dynamic RAM, 450 ns	3108003

—OR—

16K RAM List

A3	DIP Shunt	2100041
A71	DIP Shunt	2100041
Z13	16384 bit, Dynamic RAM, 450 ns	3108009
Z14	16384 bit, Dynamic RAM, 450 ns	3108009
Z15	16384 bit, Dynamic RAM, 450 ns	3108009
Z16	16384 bit, Dynamic RAM, 450 ns	3108009
Z17	16384 bit, Dynamic RAM, 450 ns	3108009
Z18	16384 bit, Dynamic RAM, 450 ns	3108009
Z19	16384 bit, Dynamic RAM, 450 ns	3108009
Z20	16384 bit, Dynamic RAM, 450 ns	3108009

Keyboard

Symbol	Description	Part Number
	Printed Circuit Board, Keyboard	1700070
	CAPACITORS	
C1	0.1 μF, 10%, 12V, Disc	1500052
C2	0.1 μF, 10%, 12V, Disc	1500052
	DIODES	
CR1	LED, HP5082–4B50, Red	2400025
	KEYBOARD	
KB1	DS5300, 53 Key, 2–Shot Key caps	5100013
	RESISTORS	
R1	4.7 K, 1/4W, 5%	4704061
R2	4.7 K, 1/4W, 5%	4704061
R3	4.7 K, 1/4W, 5%	4704061
R4	4.7 K, 1/4W, 5%	4704061
R5	4.7 K, 1/4W, 5%	4704061
R6	4.7 K, 1/4W, 5%	4704061
R7	4.7 K, 1/4W, 5%	4704061
R8	4.7 K, 1/4W, 5%	4704061
R9	330 ohm, 1/4W, 5%	4704036
	INTEGRATED CIRCUITS	
Z1	74LS05, Hex Buffer with open collector High Voltage outputs	3102009
Z2	74LS05, Hex Buffer with open collector High Voltage outputs	3102009
Z3	74LS368, TRI-STATE Hex Buffer	3102025
Z4	74LS368, TRI-STATE Hex Buffer	3102025
	WIRE	
W1	Stranded, Prebonded, LEO, Red, 10"	6002526
W2	Stranded, Prebonded, LEO, Black, 10"	6000526

Level II Kit

Symbol	Description	Part Number
	Printed Circuit Board, Level II ROM Adapter	1700081
J1	Socket, I.C., 24 Pin	2100034
R1	Resistor, 4.7K, 1/4W, 5%	4704061
Z1	I.C., 4K x 8 ROM, 450ns, ROM A	310B013
Z2	I.C., 4K x 8 ROM, 450ns, ROM B	3108014
Z3	I.C., 4K x 8 ROM, 450ns, RDM C	310B015
Z4	I.C., 74LS42, BCD to Decimal Decoder	3102036

Schematics

Schematics

BASIC I ROMs

Since the TRS-80 went into production, there have been three major PCB changes. These changes reflect different vendor's responses to system requirements as it pertains to the ROM (Read Only Memory). In the manufacturing process, certain vendors were able to supply ROMs to the factory at different times. Thus there are three major types of ROMs. There have been slight PCB Modifications as each ROM was used.

Intel EPROMs

The first mass-produced ROM type was the Intel 2616 EPROM (Erasable, Programmable, Read Only Memory). An Intel ROM may be identified by part number, the Intel trademark and the white painted crystal window. There are two Board versions that use the Intel ROM.

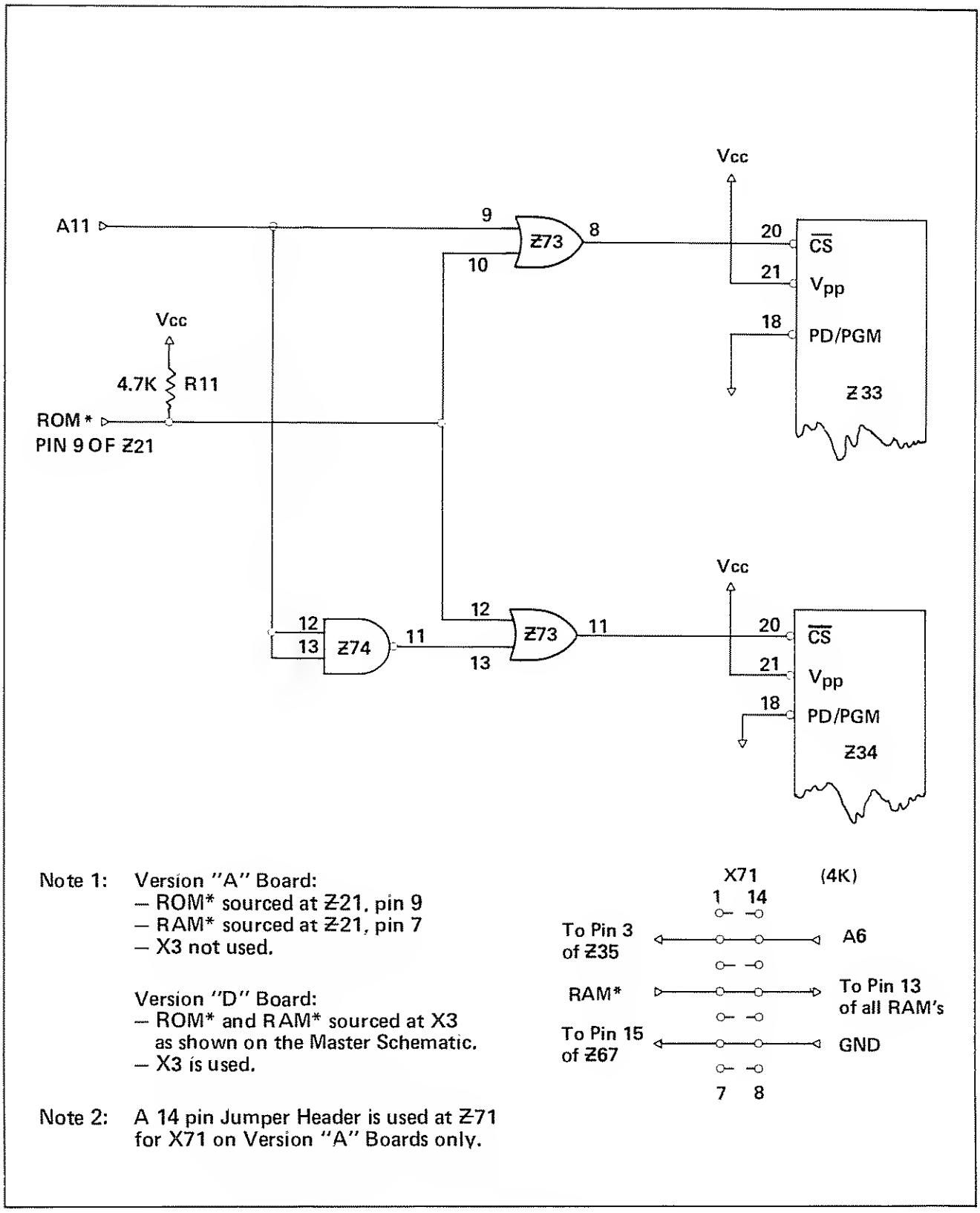
The first Board version may be identified by the "A" right after the part number on the etched side. For example, a Board marked "TRS-80 17ØØ69A" is a version "A" Board. It may also be identified by major wire modifications in two areas. With the PCB upside down and the DIN plugs facing away from you, an "A" Board has several wires on the right side between you and the DIN plugs. It also has wiring to the left of the ROMs, closest to the CPU (Central Processing Unit, the Z-8Ø). The modifications on the right side are connections in the Video, Horizontal and Vertical Sync generation areas. The wire modifications to the left are changes in the Board to use the Intel EPROMs. Whenever a Board uses the Intel EPROM set, there will always be some added circuitry associated with the $\overline{CS}$ (read "not chip select") pins for each device. This added circuitry uses spare gates on the CPU Board. Figure 23 shows the circuitry differences.

The second Board level that used Intel EPROMs was the "D" version (which may be identified by the letter "D" after the part number). For example, a Board marked "TRS-80 17ØØ69D" is a version "D" Board. It may also be identified by major wire modifications in one area. These wire modifications also utilize the spare gates on the Board for proper $\overline{CS}$ action. The wiring changes for this Board version are the same as for an "A" version Board (except some of the wiring needed on the "A" Board is not used on the "D" version because of etch pattern changes).

Since both the "A" and "D" version Boards use the Intel EPROM, you should note that each EPROM contains lettering that identifies "ROM A" or "ROM B".

ROM A must be in Z33's socket; ROM B must be in Z34's socket. No exceptions!

Also, you should note that both Board versions have etch cuts. The "A" Board has many visible etch cuts while the "D" has only a few. Do not try to "correct" these cuts. The Board will not operate without them.



Note 1: Version "A" Board:
- ROM* sourced at Z21, pin 9
- RAM* sourced at Z21, pin 7
- X3 not used.

Version "D" Board:
- ROM* and RAM* sourced at X3 as shown on the Master Schematic.
- X3 is used.

Note 2: A 14 pin Jumper Header is used at Z71 for X71 on Version "A" Boards only.

FIGURE 23. Spare Gate Usage on Version "A" or "D" Boards

National MM2316 ROMs

The National ROMs were the second type of ROM the factory received during production. These ROMs were used in a large number of "D" version Boards and in a few "A" version Boards. These devices are true Read Only Memories since they are not programmable (except by the vendor), nor are they eraseable. They may be identified by the colored ceramic package (only a few) or in 24 pin dual-in-line plastic packages. The ceramic packages have the part number MM2316 followed by either an "R/D" or an "S/D". The "R/D" is ROM A and the "S/D" is ROM B. The plastic packaged ROM has the part number "M2316E" and below it "MMS25BET" followed by an "R/N" or an "S/N". Once again, the "R/N" is ROM A, while the "S/N" is ROM B.

These ROMs may be in either Z33 or Z34 socket and still operate correctly. In other words, ROM A does not have to be in socket Z33. It could be in socket Z34, and vice versa for ROM B. The reason you need to know the differences between ROM A and ROM B is in troubleshooting. Certain software troubleshooting aids may fail ROM A and pass ROM B. You need to identify ROM A to replace it or for further troubleshooting.

NOTE: The only differences between the Schematic on this page and the Master Schematic are that A11 is on Pin 20 and ROM is on Pin 18.

The "D" version Board with National ROMs will usually have only two wire modifications or jumpers, directly under the ROM sockets. These jumpers will also have two etch cuts associated with them. Version "A" Boards using National ROMs may have more jumpers. This is because an "A" version Board may have been modified to use Intel EPROMs and then later remodified to accept the National ROMs. Be careful when identifying the different Board versions. The two wire modifications associated with National ROMs are shown in Figure 24. Notice that only two wires are switched around. There are no spare gates used in this modification.

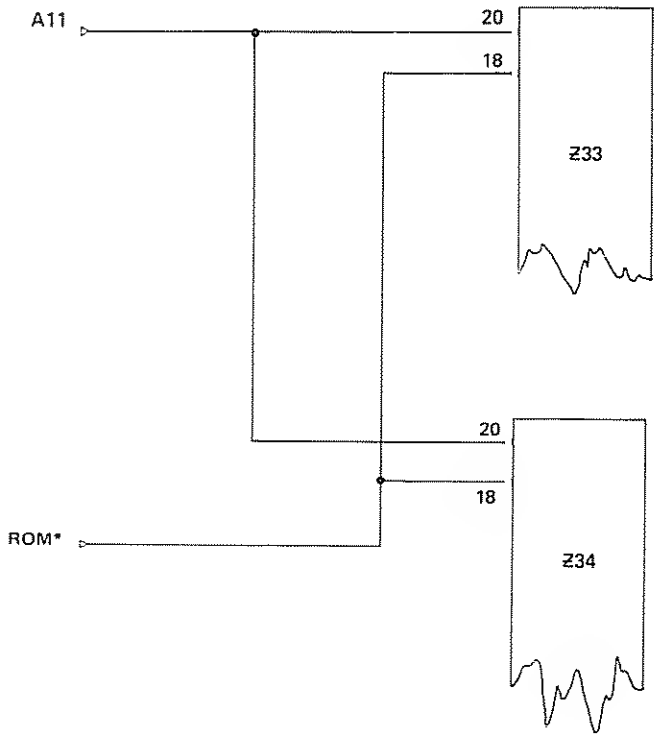


FIGURE 24. National ROMs in Version "D" Board*

Motorola 7800 Series ROMs (Two Chip Set)

Motorola was the next vendor to supply ROMs for the TRS-80. These ROMs may be identified by the part number 7BØ7 for ROM A and 78Ø4 for ROM B. These devices are used only on "D" version Boards and there are no PCB modifications. As with National ROMs, Motorola ROMs may be placed in either Z33's or Z34's socket.

Motorola 7800 Series ROMs (Single Chip Set)

The last ROM supplied to the factory was a single Motorola ROM. It may be packaged in either ceramic (like a few of the National's) or in plastic. The device's number is "7BØ9" and also says "BASIC I". This chip may be inserted in either Z33 or Z34, but it's usually put in Z33. Once again, there are no PCB modifications. Notice the part number on the single chip ROM and the 78Ø9 ROM B mentioned above. Be very careful when replacing ROMs.

BASIC II ROMs

The TRS-80 having BASIC II ROMs are easily identified. There are no ROMs plugged into Z33 or Z34. Instead, there is a flat ribbon cable interconnecting the CPU Board to a small 4-chip ROM Board. This ROM Board is attached with double-sided tape to the etched side of the CPU Board. The three ROMs on this Board contain BASIC II. These ROMs may be supplied by various vendors. Figure 25 is the Schematic for Level II BASIC.

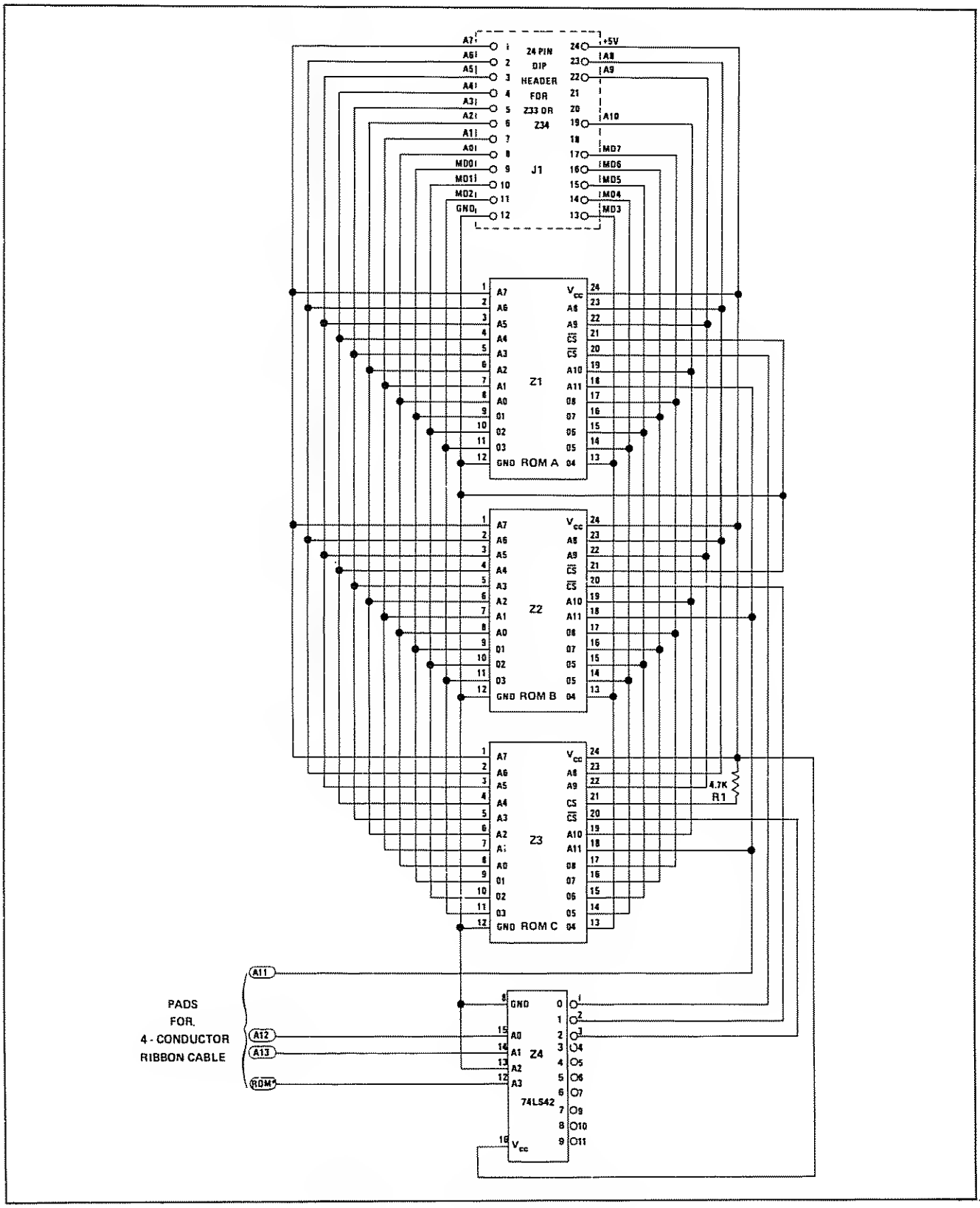
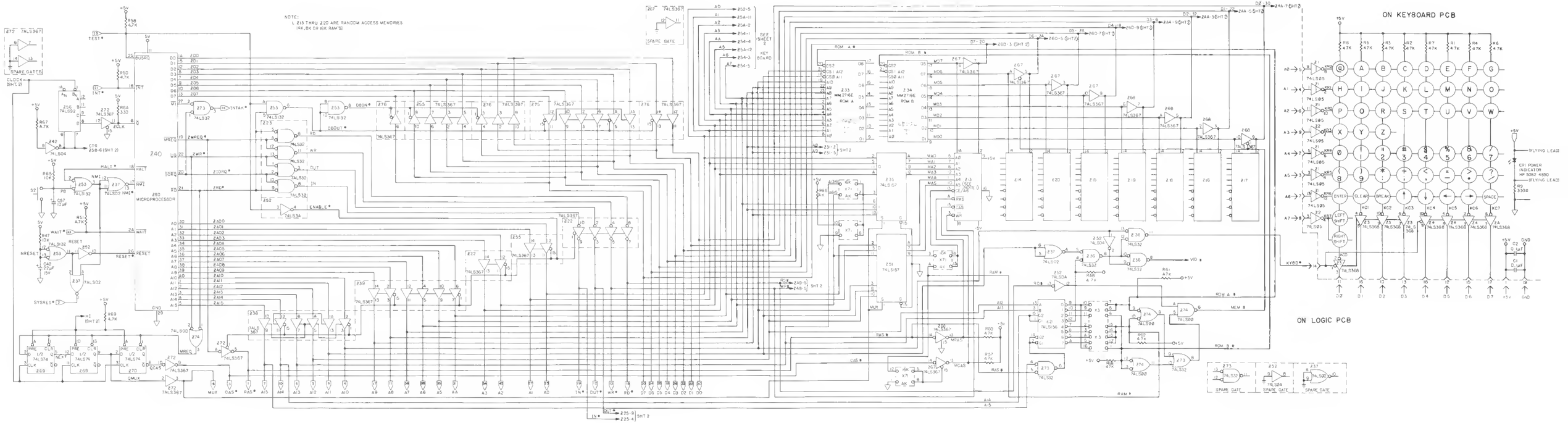
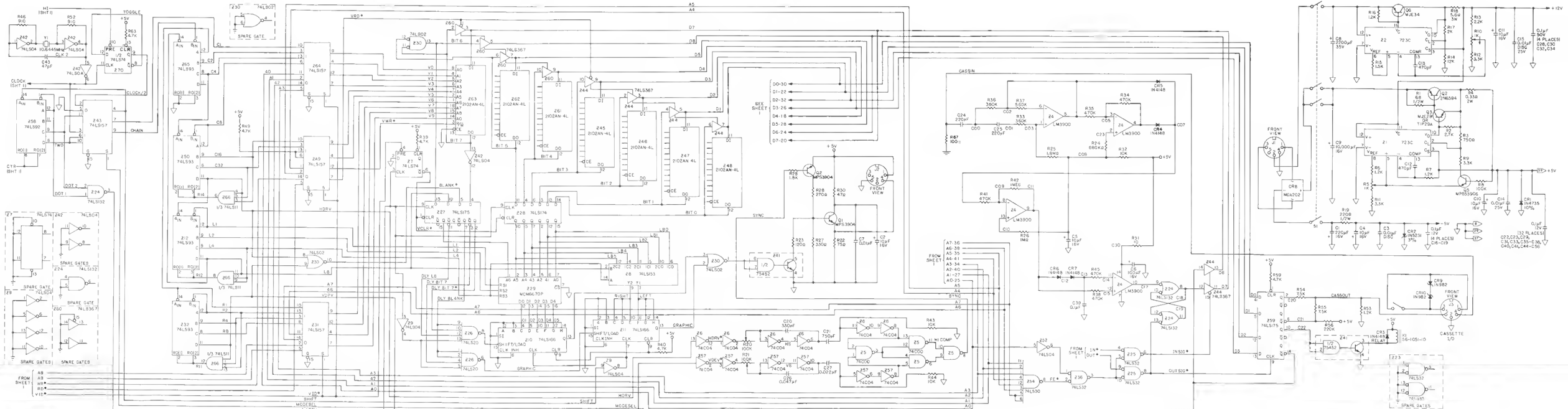


FIGURE 25. Level II BASIC Schematic



TRS-80 Schematic (Sheet 1)



TRS-80 Schematic (Sheet 2)

RADIO SHACK A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U K
280-316 VICTORIA ROAD RYDALMERE N S W 2116	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

PRINTED IN U.S.A.